JUNIPER NETWORKS | Engineering Simplicity

Junos® OS

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Published
2023-12-15

JUNOS

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

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

Junos® OS Routing Policies, Firewall Filters, and Traffic Policers User Guide
Copyright © 2023 Juniper Networks, Inc. All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About This Guide | xxxvi

1 **Understanding and Configuring Junos Routing Policies**

Overview | 2

Policy Framework Overview | **2**

Comparison of Routing Policies and Firewall Filters | **9**

Prefix Prioritization Overview | **15**

FIB Prefix Prioritization | **16**

Accounting of the Policer Overhead Attribute at the Interface Level | **17**

Configuring the Accounting of Policer Overhead in Interface Statistics | **19**

Understanding Routing Policies | **22**

Protocol Support for Import and Export Policies | **26**

Example: Applying Routing Policies at Different Levels of the BGP Hierarchy | **27**

- Requirements | **27**
- Overview | **28**
- Configuration | **30**
- Verification | **36**

Default Routing Policies | **40**

Example: Configuring a Conditional Default Route Policy | **44**

- Requirements | **44**
- Overview | **44**
- Configuration | **45**
- Verification | **50**

Evaluating Routing Policies Using Match Conditions, Actions, Terms, and Expressions | 54

How a Routing Policy Is Evaluated | **54**

Categories of Routing Policy Match Conditions | **56**

Routing Policy Match Conditions | **58**

Route Filter Match Conditions | **72**

Actions in Routing Policy Terms | **75**

Summary of Routing Policy Actions | **92**

Example: Configuring a Routing Policy to Advertise the Best External Route to Internal Peers | **95**

- Requirements | **97**
- Overview | **97**
- Configuration | **99**
- Verification | **103**

Example: Configuring BGP to Advertise Inactive Routes | **107**

- Requirements | **109**
- Overview | **109**
- Configuration | **110**
- Verification | **114**

Example: Using Routing Policy to Set a Preference Value for BGP Routes | **118**

- Requirements | **118**
- Overview | **118**
- Configuration | **120**
- Verification | **125**

Example: Enabling BGP Route Advertisements | **126**

- Requirements | **127**
- Overview | **127**
- Configuration | **128**
- Verification | **134**

Example: Rejecting Known Invalid Routes | **137**

- Requirements | **138**
- Overview | **138**
- Configuration | **138**
- Verification | **140**

Example: Using Routing Policy in an ISP Network | **141**

- Requirements | **141**
- Overview | **141**
- Set Commands for All Devices in the Topology | **144**

Configuring Device Customer-1 | **152**

Configuring Device Customer-2 | **155**

Configuring Devices ISP-1 and ISP-2 | **159**

Configuring Device ISP-3 | **166**

Configuring Device Exchange-2 | **173**

Configuring Device Private-Peer-2 | **176**

Verification | **182**

Understanding Policy Expressions | **204**

Understanding Backup Selection Policy for OSPF Protocol | **210**

Configuring Backup Selection Policy for the OSPF Protocol | **211**

Configuring Backup Selection Policy for IS-IS Protocol | **219**

Understanding Backup Selection Policy for IS-IS Protocol | **219**

Example: Configuring Backup Selection Policy for the OSPF or OSPF3 Protocol | **221**

Requirements | **221**

Overview | **222**

Configuration | **223**

Verification | **248**

Evaluating Complex Cases Using Policy Chains and Subroutines | 256

Understanding How a Routing Policy Chain Is Evaluated | **256**

Example: Configuring Policy Chains and Route Filters | **258**

Requirements | **258**

Overview | **258**

Configuration | **261**

Verification | **270**

Example: Using Firewall Filter Chains | **275**

Requirements | **275**

Overview | **276**

Configuration | **277**

Verification | **282**

Understanding Policy Subroutines in Routing Policy Match Conditions | **283**

How a Routing Policy Subroutine Is Evaluated | **287**

Example: Configuring a Policy Subroutine | 289

- Requirements | 289
- Overview | 289
- Configuration | 292
- Verification | 299

Configuring Route Filters and Prefix Lists as Match Conditions | 303

Understanding Route Filters for Use in Routing Policy Match Conditions | 303

Understanding Route Filter and Source Address Filter Lists for Use in Routing Policy Match Conditions | 327

Understanding Load Balancing Using Source or Destination IP Only | 327

Configuring Load Balancing Using Source or Destination IP Only | 328

Walkup for Route Filters Overview | 330

Configuring Walkup for Route Filters to Improve Operational Efficiency | 334

Example: Configuring Route Filter Lists | 340

- Requirements | 340
- Overview | 340
- Configuration | 341
- Verification | 343

Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency | 346

- Requirements | 346
- Overview | 347
- Configuring Route Filter Walkup Globally | 348
- Verification | 352
- Troubleshooting | 353

Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency | 354

- Requirements | 355
- Overview | 355
- Configuring Route Filter Walkup Locally | 356
- Verification | 360
- Troubleshooting | 361

Example: Configuring a Route Filter Policy to Specify Priority for Prefixes Learned Through OSPF | 362

Requirements | **362**

Overview | **363**

Configuration | **364**

Verification | **368**

Example: Configuring the MED Using Route Filters | **368**

Requirements | **369**

Overview | **369**

Configuration | **370**

Verification | **385**

Example: Configuring Layer 3 VPN Protocol Family Qualifiers for Route Filters | **388**

Requirements | **388**

Overview | **388**

Configuration | **389**

Verification | **392**

Understanding Prefix Lists for Use in Routing Policy Match Conditions | **392**

Example: Configuring Routing Policy Prefix Lists | **396**

Requirements | **397**

Overview | **397**

Configuration | **400**

Verification | **408**

Example: Configuring the Priority for Route Prefixes in the RPD Infrastructure | **412**

Requirements | **412**

Overview | **413**

Configuration | **414**

Verification | **421**

Configuring Priority for Route Prefixes in RPD Infrastructure | **426**

Configuring AS Paths as Match Conditions | 433

Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions | **433**

Example: Using AS Path Regular Expressions | **442**

Requirements | **442**

Overview | **443**

Configuration | **445**

Verification | 453

Understanding Prepending AS Numbers to BGP AS Paths | 457

Example: Configuring a Routing Policy for AS Path Prepending | 458

Requirements | 458

Overview | 458

Configuration | 459

Verification | 464

Appendix Full Configurations | 467

Understanding Adding AS Numbers to BGP AS Paths | 468

Example: Advertising Multiple Paths in BGP | 469

Requirements | 470

Overview | 470

Configuration | 472

Verification | 499

Improve the Performance of AS Path Lookup in BGP Policy | 506

AS Path Lookup in a BGP Policy Without Regular Expression Overview | 507

Configure AS Path Lookup Without Using Regular Expression | 508

Configuring Communities as Match Conditions | 511

Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions | 511

Understanding How to Define BGP Communities and Extended Communities | 513

How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions | 521

Example: Configuring Communities in a Routing Policy | 528

Requirements | 528

Overview | 529

Configuration | 531

Verification | 543

Example: Configuring Extended Communities in a Routing Policy | 548

Requirements | 548

Overview | 548

Configuration | 550

Verification | 555

Example: Configuring BGP Large Communities | 560

Requirements | 560

Overview | 560

Configuration | 561

Verification | 567

Example: Configuring a Routing Policy Based on the Number of BGP Communities | 572

Requirements | 572

Overview | 572

Configuration | 573

Verification | 580

Example: Configuring a Routing Policy That Removes BGP Communities | 583

Requirements | 583

Overview | 583

Configuration | 585

Verification | 592

Increasing Network Stability with BGP Route Flapping Actions | 596

Understanding Damping Parameters | 596

Using Routing Policies to Damp BGP Route Flapping | 598

Example: Configuring BGP Route Flap Damping Parameters | 604

Requirements | 605

Overview | 605

Configuration | 606

Verification | 612

Example: Configuring BGP Route Flap Damping Based on the MBGP MVPN Address Family | 619

Requirements | 619

Overview | 619

Configuration | 620

Verification | 632

Tracking Traffic Usage with Source Class Usage and Destination Class Usage Actions | 635

Understanding Source Class Usage and Destination Class Usage Options | 635

Source Class Usage Overview | **637**

Guidelines for Configuring SCU | **638**

System Requirements for SCU | **639**

Terms and Acronyms for SCU | **640**

- destination class usage (DCU) | **640**
- source class usage (SCU) | **641**
- source address (SA) | **641**
- destination address (DA) | **641**

Roadmap for Configuring SCU | **641**

Roadmap for Configuring SCU with Layer 3 VPNs | **642**

Configuring Route Filters and Source Classes in a Routing Policy | **642**

Applying the Policy to the Forwarding Table | **644**

Enabling Accounting on Inbound and Outbound Interfaces | **644**

Configuring Input SCU on the vt Interface of the Egress PE Router | **645**

Mapping the SCU-Enabled vt Interface to the VRF Instance | **646**

Configuring SCU on the Output Interface | **647**

Associating an Accounting Profile with SCU Classes | **648**

Verifying Your SCU Accounting Profile | **649**

SCU Configuration | **650**

SCU with Layer 3 VPNs Configuration | **661**

Example: Grouping Source and Destination Prefixes into a Forwarding Class | **671**

- Requirements | **672**
- Overview | **672**
- Configuration | **675**
- Verification | **682**

Avoiding Traffic Routing Threats with Conditional Routing Policies | 686

Conditional Advertisement and Import Policy (Routing Table) with certain match conditions | **686**

Conditional Advertisement Enabling Conditional Installation of Prefixes Use Cases | **689**

Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table | **691**

- Requirements | **691**
- Overview | **691**
- Configuration | **695**
- Verification | **705**

Protecting Against DoS Attacks by Forwarding Traffic to the Discard Interface | 714

Assigning Forwarding Classes and Loss Priority | **714**

Understanding Forwarding Packets to the Discard Interface | **716**

Example: Forwarding Packets to the Discard Interface | **718**

- Requirements | **718**
- Overview | **718**
- Configuration | **722**
- Verification | **727**

Improving Commit Times with Dynamic Routing Policies | 732

Understanding Dynamic Routing Policies | **732**

Example: Configuring Dynamic Routing Policies | **737**

- Requirements | **737**
- Overview | **737**
- Configuration | **739**
- Verification | **750**

Testing Before Applying Routing Policies | 758

Understanding Routing Policy Tests | **758**

Example: Testing a Routing Policy with Complex Regular Expressions | **760**

- Requirements | **760**
- Overview | **760**
- Configuration | **763**
- Verification | **769**

2 **Configuring Firewall Filters**

Understanding How Firewall Filters Protect Your Network | 772

Firewall Filters Overview | **772**

Router Data Flow Overview | 773

Stateless Firewall Filter Overview | 776

Understanding How to Use Standard Firewall Filters | 778

Understanding How Firewall Filters Control Packet Flows | 779

Stateless Firewall Filter Components | 781

Stateless Firewall Filter Application Points | 788

How Standard Firewall Filters Evaluate Packets | 793

Understanding Firewall Filter Fast Lookup Filter | 798

Understanding Egress Firewall Filters with PVLANs | 799

Selective Class-based Filtering on PTX Routers | 800

- Selective Class-based Filtering on PTX Routers | 800
- Understanding Class-based Filtering on PTX Routers | 802
- Example: Selective Class Based Filtering (PTX Routers) | 803

Guidelines for Configuring Firewall Filters | 814

Guidelines for Applying Standard Firewall Filters | 821

Supported Standards for Filtering | 826

Monitoring Firewall Filter Traffic | 827

- Monitoring Traffic for All Firewall Filters and Policers That Are Configured | 827
- Monitoring Traffic for a Specific Firewall Filter | 828
- Monitoring Traffic for a Specific Policer | 829

Troubleshooting Firewall Filters | 830

- Troubleshooting QFX10000 Switches | 831
  - Do Not Combine Match Conditions for Different Layers | 831
  - Layer 2 Packets Cannot be Discarded with Firewall Filters | 831
  - Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces | 832
- Troubleshooting Other Switches | 832
  - Firewall Filter Configuration Returns a No Space Available in TCAM Message | 833
  - Filter Counts Previously Dropped Packet | 836
  - Matching Packets Not Counted | 837
  - Counter Reset When Editing Filter | 838

Cannot Include loss-priority and policer Actions in Same Term | **838**

Cannot Egress Filter Certain Traffic Originating on QFX Switch | **839**

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **840**

Egress Firewall Filters with Private VLANs | **840**

Egress Filtering of L2PT Traffic Not Supported | **842**

Cannot Drop BGP Packets in Certain Circumstances | **842**

Invalid Statistics for Policer | **843**

Policers can Limit Egress Filters | **843**

Firewall Filter Match Conditions and Actions | 845

Overview of Firewall Filters (OCX Series) | **846**

Understanding Firewall Filter Match Conditions | **848**

Understanding Firewall Filter Planning | **853**

Understanding How Firewall Filters Are Evaluated | **854**

Understanding Firewall Filter Match Conditions | **856**

Firewall Filter Flexible Match Conditions | **862**

Firewall Filter Nonterminating Actions | **871**

Firewall Filter Terminating Actions | **884**

Firewall Filter Match Conditions and Actions (ACX Series Routers) | **909**

Overview of Firewall Filter Match Conditions and Actions on ACX Series Routers | **910**

Match Conditions for Bridge Family Firewall Filters (ACX Series Routers) | **912**

Match Conditions for CCC Firewall Family Filters (ACX Series Routers) | **916**

Match Conditions for IPv4 Traffic (ACX Series Routers) | **917**

Match Conditions for IPv6 Traffic (ACX Series Routers) | **923**

Match Conditions for MPLS Traffic (ACX Series Routers) | **930**

Nonterminating Actions (ACX Series Routers) | **931**

Terminating Actions (ACX Series Routers) | **935**

Firewall Filter Match Conditions for Protocol-Independent Traffic | **937**

Firewall Filter Match Conditions for IPv4 Traffic | **940**

Firewall Filter Match Conditions for IPv6 Traffic | **957**

Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**

Firewall Filter Match Conditions Based on Bit-Field Values | **976**

Firewall Filter Match Conditions Based on Address Fields | **982**

Firewall Filter Match Conditions Based on Address Classes | **992**

Understanding IP-Based Filtering and Selective Port Mirroring of MPLS Traffic | **994**

Firewall Filter Match Conditions for MPLS Traffic | **1000**

Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic | **1009**

Firewall Filter Match Conditions for VPLS Traffic | **1013**

Firewall Filter Match Conditions for Layer 2 CCC Traffic | **1032**

Firewall Filter Match Conditions for Layer 2 Bridging Traffic | **1037**

Firewall Filter Support on Loopback Interface | **1055**

Applying Firewall Filters to Routing Engine Traffic | 1057

Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs | **1057**

Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List | **1059**

- Requirements | **1059**
- Overview | **1060**
- Configuration | **1060**
- Verification | **1063**

Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | **1065**

- Requirements | **1065**
- Overview | **1065**
- Configuration | **1066**
- Verification | **1069**

Example: Configure a Filter to Block Telnet and SSH Access | **1073**

- Requirements | **1073**
- Overview and Topology | **1073**
- Configuration | **1074**
- Verify the Stateless Firewall Filter | **1082**

Example: Configuring a Filter to Block TFTP Access | **1086**

- Requirements | **1086**

Overview | **1086**

Configuration | **1087**

Verification | **1090**

Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags | **1091**

Requirements | **1091**

Overview | **1091**

Configuration | **1091**

Verification | **1094**

Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers | **1095**

Requirements | **1095**

Overview | **1095**

Configuration | **1096**

Verification | **1101**

Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods | **1103**

Requirements | **1104**

Overview | **1104**

Configuration | **1105**

Verification | **1112**

Example: Protecting the Routing Engine with a Packets-Per-Second Rate Limiting Filter | **1119**

Requirements | **1120**

Overview | **1120**

Configuration | **1120**

Verification | **1123**

Example: Configuring a Filter to Exclude DHCPv6 and ICMPv6 Control Traffic for LAC Subscriber | **1124**

Requirements | **1125**

Overview | **1125**

Configuration | **1125**

Port Number Requirements for DHCP Firewall Filters | **1131**

Example: Configuring a DHCP Firewall Filter to Protect the Routing Engine | **1132**

Requirements | **1132**

Overview | **1132**

Configuration | **1133**

Verification | 1136

Applying Firewall Filters to Transit Traffic | 1138

Example: Configuring a Filter for Use as an Ingress Queuing Filter | 1138

Requirements | 1139

Overview | 1139

Configuration | 1140

Example: Configuring a Filter to Match on IPv6 Flags | 1142

Requirements | 1142

Overview | 1142

Configuration | 1142

Verification | 1144

Example: Configuring a Filter to Match on Port and Protocol Fields | 1144

Requirements | 1144

Overview | 1144

Configuration | 1144

Verification | 1149

Example: Configuring a Filter to Count Accepted and Rejected Packets | 1149

Requirements | 1149

Overview | 1149

Configuration | 1150

Verification | 1153

Example: Configuring a Filter to Count and Discard IP Options Packets | 1154

Requirements | 1154

Overview | 1154

Configuration | 1155

Verification | 1158

Example: Configuring a Filter to Count IP Options Packets | 1158

Requirements | 1159

Overview | 1159

Configuration | 1159

Verification | 1165

Example: Configuring a Filter to Count and Sample Accepted Packets | 1165

Requirements | **1166**
Overview | **1166**
Configuration | **1167**
Verification | **1170**

Example: Configuring a Filter to Set the DSCP Bit to Zero | **1172**

Requirements | **1173**
Overview | **1173**
Configuration | **1173**
Verification | **1176**

Example: Configuring a Filter to Set the DSCP Bit to Zero | **1177**

Requirements | **1177**
Overview | **1177**
Configuration | **1177**
Verification | **1180**

Example: Configuring a Filter to Match on Two Unrelated Criteria | **1181**

Requirements | **1181**
Overview | **1181**
Configuration | **1181**
Verification | **1184**

Example: Configuring a Filter to Accept DHCP Packets Based on Address | **1185**

Requirements | **1185**
Overview | **1185**
Configuration | **1185**
Verification | **1188**

Example: Configuring a Filter to Accept OSPF Packets from a Prefix | **1189**

Requirements | **1189**
Overview | **1189**
Configuration | **1189**
Verification | **1193**

Example: Configuring a Stateless Firewall Filter to Handle Fragments | **1193**

Requirements | **1193**
Overview | **1194**

Configuration | **1195**

Verification | **1199**

Configuring a Firewall Filter to Prevent or Allow IPv4 Packet Fragmentation | **1201**

Configuring a Firewall Filter to Discard Ingress IPv6 Packets with a Mobility Extension Header | **1202**

Example: Configuring an Egress Filter Based on IPv6 Source or Destination IP Addresses | **1203**

Requirements | **1203**

Overview | **1203**

Configuration | **1204**

Example: Configuring a Rate-Limiting Filter Based on Destination Class | **1208**

Requirements | **1208**

Overview | **1208**

Configuration | **1208**

Verification | **1212**

Configuring Firewall Filters in Logical Systems | 1213

Firewall Filters in Logical Systems Overview | **1213**

Guidelines for Configuring and Applying Firewall Filters in Logical Systems | **1215**

References from a Firewall Filter in a Logical System to Subordinate Objects | **1218**

References from a Firewall Filter in a Logical System to Nonfirewall Objects | **1220**

References from a Nonfirewall Object in a Logical System to a Firewall Filter | **1223**

Example: Configuring Filter-Based Forwarding | **1230**

Requirements | **1230**

Overview | **1230**

Configuration | **1231**

Example: Configuring Filter-Based Forwarding on Logical Systems | **1236**

Requirements | **1236**

Overview | **1237**

Configuration | **1240**

Verification | **1247**

Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | **1250**

Requirements | **1251**

Overview | **1251**

Configuration | **1252**

Verification | **1255**

Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | **1256**

Requirements | **1257**

Overview | **1257**

Configuration | **1258**

Verification | **1261**

Unsupported Firewall Filter Statements for Logical Systems | **1262**

Unsupported Actions for Firewall Filters in Logical Systems | **1265**

Filter-Based Forwarding for Routing Instances | **1272**

Forwarding Table Filters for Routing Instances on ACX Series Routers | **1273**

Configuring Forwarding Table Filters | **1274**

Configuring Firewall Filter Accounting and Logging | 1277

Accounting for Firewall Filters Overview | **1277**

System Logging Overview | **1278**

System Logging of Events Generated for the Firewall Facility | **1279**

Firewall Filter Logging Actions | **1282**

Example: Configuring Statistics Collection for a Firewall Filter | **1286**

Requirements | **1286**

Overview | **1286**

Configuration | **1287**

Verification | **1293**

Example: Configuring Logging for a Firewall Filter Term | **1293**

Requirements | **1294**

Overview | **1294**

Configuration | **1294**

Verification | **1298**

Attaching Multiple Firewall Filters to a Single Interface | 1300

Applying Firewall Filters to Interfaces | **1300**

Configuring Firewall Filters | **1301**

- Configuring a Firewall Filter | **1301**
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | **1303**

Multifield Classifier Example: Configuring Multifield Classification | **1304**

- Multifield Classification Overview | **1304**
- Multifield Classification Requirements and Restrictions | **1307**
- Multifield Classification Limitations on M Series Routers | **1308**
- Example: Configuring Multifield Classification | **1311**
  - Requirements | **1311**
  - Overview | **1312**
  - Configuration | **1313**
  - Verification | **1319**
- Example: Configuring and Applying a Firewall Filter for a Multifield Classifier | **1321**
  - Requirements | **1321**
  - Overview | **1321**
  - Configuration | **1323**
  - Verification | **1327**

Multifield Classifier for Ingress Queuing on MX Series Routers with MPC | **1330**

Assigning Multifield Classifiers in Firewall Filters to Specify Packet-Forwarding Behavior (CLI Procedure) | **1331**

Understanding Multiple Firewall Filters in a Nested Configuration | **1334**

Guidelines for Nesting References to Multiple Firewall Filters | **1336**

Understanding Multiple Firewall Filters Applied as a List | **1338**

Guidelines for Applying Multiple Firewall Filters as a List | **1342**

Example: Applying Lists of Multiple Firewall Filters | **1344**

- Requirements | **1345**
- Overview | **1345**
- Configuration | **1346**
- Verification | **1351**

Example: Nesting References to Multiple Firewall Filters | **1352**

- Requirements | **1352**
- Overview | **1353**

Configuration | **1353**

Verification | **1357**

Example: Filtering Packets Received on an Interface Set | **1358**

Requirements | **1358**

Overview | **1358**

Configuration | **1359**

Verification | **1367**

Attaching a Single Firewall Filter to Multiple Interfaces | 1368

Interface-Specific Firewall Filter Instances Overview | **1368**

Interface-Specific Firewall Filter Instances Overview | **1371**

Filtering Packets Received on a Set of Interface Groups Overview | **1373**

Filtering Packets Received on an Interface Set Overview | **1374**

Example: Configuring Interface-Specific Firewall Filter Counters | **1375**

Requirements | **1375**

Overview | **1375**

Configuration | **1376**

Verification | **1380**

Example: Configuring a Stateless Firewall Filter on an Interface Group | **1382**

Requirements | **1382**

Overview | **1382**

Configuration | **1383**

Verification | **1388**

Configuring Filter-Based Tunneling Across IP Networks | 1394

Understanding Filter-Based Tunneling Across IPv4 Networks | **1394**

Firewall Filter-Based L2TP Tunneling in IPv4 Networks Overview | **1397**

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | **1401**

Components of Filter-Based Tunneling Across IPv4 Networks | **1403**

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | **1409**

Requirements | **1409**

Overview | **1411**

Configuration | **1414**

Verification | **1425**

Configuring Service Filters | 1431

Service Filter Overview | **1431**

How Service Filters Evaluate Packets | **1433**

Guidelines for Configuring Service Filters | **1435**

Guidelines for Applying Service Filters | **1438**

Example: Configuring and Applying Service Filters | **1441**

Requirements | **1441**

Overview | **1442**

Configuration | **1443**

Verification | **1447**

Service Filter Match Conditions for IPv4 or IPv6 Traffic | **1449**

Service Filter Nonterminating Actions | **1460**

Service Filter Terminating Actions | **1461**

Configuring Simple Filters | 1463

Simple Filter Overview | **1463**

How Simple Filters Evaluate Packets | **1464**

Guidelines for Configuring Simple Filters | **1465**

Guidelines for Applying Simple Filters | **1470**

Example: Configuring and Applying a Simple Filter | **1471**

Requirements | **1471**

Overview | **1472**

Configuration | **1472**

Verification | **1476**

Configuring Layer 2 Firewall Filters | 1479

Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | **1479**

Example: Configuring Filtering of Frames by MAC Address | **1480**

Example: Configuring Filtering of Frames by IEEE 802.1p Bits | **1482**

Example: Configuring Filtering of Frames by Packet Loss Priority | **1483**

Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | **1485**

Understanding Firewall Filters on OVSDB-Managed Interfaces | **1488**

Example: Applying a Firewall Filter to OVSDB-Managed Interfaces | **1489**

- Requirements | **1490**
- Overview | **1490**
- Configuration | **1490**

Configuring Firewall Filters for Forwarding, Fragments, and Policing | 1493

Filter-Based Forwarding Overview | **1493**

Firewall Filters That Handle Fragmented Packets Overview | **1496**

Stateless Firewall Filters That Reference Policers Overview | **1496**

Example: Configuring Filter-Based Forwarding on the Source Address | **1497**

- Requirements | **1498**
- Overview | **1498**
- Configuration | **1501**
- Verification | **1509**

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**

- Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**
- Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface | **1513**
  - Requirements | **1514**
  - Overview | **1514**
  - Configuration | **1515**
  - Verification | **1519**
- Example: Configuring Filter-Based Forwarding to a Specific Destination IP Address | **1520**
  - Requirements | **1521**
  - Overview | **1521**
  - Configuration | **1522**
  - Verification | **1531**

Configuring Firewall Filters (EX Series Switches) | 1534

Firewall Filters for EX Series Switches Overview | **1535**

Understanding Planning of Firewall Filters | **1539**

Understanding Firewall Filter Match Conditions | **1543**

Understanding How Firewall Filters Control Packet Flows | **1549**

Understanding How Firewall Filters Are Evaluated | **1550**

Understanding Firewall Filter Processing Points for Bridged and Routed Packets on EX Series Switches | **1552**

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**

Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches | **1570**

Support for Match Conditions and Actions for Loopback Firewall Filters on Switches | **1633**

Configuring Firewall Filters (CLI Procedure) | **1638**

- Configuring a Firewall Filter | **1638**
- Configuring a Term Specifically for IPv4 or IPv6 Traffic | **1643**
- Applying a Firewall Filter to a Port on a Switch | **1644**
- Applying a Firewall Filter to a Management Interface on a Switch | **1645**
- Applying a Firewall Filter to a VLAN on a Network | **1646**
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | **1648**

Understanding How Firewall Filters Test a Packet's Protocol | **1649**

Understanding Filter-Based Forwarding for EX Series Switches | **1650**

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

- Requirements | **1651**
- Overview | **1651**
- Configuring an Ingress Port Firewall Filter to Prioritize Voice Traffic and Rate-Limit TCP and ICMP Traffic | **1656**
- Configuring a VLAN Ingress Firewall Filter to Prevent Rogue Devices from Disrupting VoIP Traffic | **1664**
- Configuring a VLAN Firewall Filter to Count, Monitor, and Analyze Egress Traffic on the Employee VLAN | **1668**
- Configuring a VLAN Firewall Filter to Restrict Guest-to-Employee Traffic and Peer-to-Peer Applications on the Guest VLAN | **1671**
- Configuring a Router Firewall Filter to Give Priority to Egress Traffic Destined for the Corporate Subnet | **1674**
- Verification | **1676**

Example: Configuring a Firewall Filter on a Management Interface on an EX Series Switch | **1680**

Requirements | **1680**

Overview and Topology | **1680**

Configuration | **1681**

Verification | **1684**

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**

Requirements | **1686**

Overview and Topology | **1686**

Configuration | **1686**

Verification | **1690**

Example: Applying Firewall Filters to Multiple Supplicants on Interfaces Enabled for 802.1X or MAC RADIUS Authentication | **1692**

Requirements | **1692**

Overview and Topology | **1693**

Configuration | **1695**

Verification | **1698**

Verifying That Policers Are Operational | **1699**

Troubleshooting Firewall Filters | **1700**

Troubleshooting QFX10000 Switches | **1701**

Do Not Combine Match Conditions for Different Layers | **1701**

Layer 2 Packets Cannot be Discarded with Firewall Filters | **1701**

Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces | **1702**

Troubleshooting Other Switches | **1702**

Firewall Filter Configuration Returns a No Space Available in TCAM Message | **1703**

Filter Counts Previously Dropped Packet | **1706**

Matching Packets Not Counted | **1707**

Counter Reset When Editing Filter | **1708**

Cannot Include loss-priority and policer Actions in Same Term | **1708**

Cannot Egress Filter Certain Traffic Originating on QFX Switch | **1709**

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **1710**

Egress Firewall Filters with Private VLANs | **1710**

Egress Filtering of L2PT Traffic Not Supported | **1712**

Cannot Drop BGP Packets in Certain Circumstances | **1712**

Invalid Statistics for Policer | **1713**

Policers can Limit Egress Filters | **1713**

Configuring Firewall Filters (QFX Series Switches, EX4600 Switches, PTX Series Routers) | 1715

Overview of Firewall Filters (QFX Series) | **1716**

Understanding Firewall Filter Planning | **1719**

Planning the Number of Firewall Filters to Create | **1721**

- How to Increase the Number of Firewall Filters | **1721**
- TCAM | **1722**
- Avoid Configuring too Many Filters | **1723**
- Configuring TCAM Error Messages | **1723**
- How to Increase the Scale of Firewall Filters Using Profiles | **1723**
- How Policers can Limit Egress Filters | **1729**
- Planning for Filter-Specific Policers | **1730**
- Planning for Filter-Based Forwarding | **1730**

Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | **1731**

- Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700) | **1732**
- Firewall Filter Match Conditions and Actions (QFX5220 and the QFX5130-32CD) | **1762**

Firewall Filter Match Conditions and Actions (QFX10000 Switches) | **1780**

Firewall Filter Match Conditions and Actions (PTX Series Routers) | **1798**

- Firewall Filter Match Conditions and Actions (PTX Series Routers) | **1798**
- IPv6 Firewall Filter Match Conditions and Actions (PTX10001-20C) | **1815**

Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers | **1823**

Configuring Firewall Filters | **1826**

- Configuring a Firewall Filter | **1826**
- Configuring Enhanced Egress Firewall Filters (QFX5110 and QFX5220 Switches) | **1830**
- Applying a Firewall Filter to a Port | **1831**
- Applying a Firewall Filter to a VLAN | **1832**
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | **1832**
- Applying a Firewall Filter to a Layer 2 CCC (QFX10000 Switches) | **1834**

Applying Firewall Filters to Interfaces | **1835**

Overview of MPLS Firewall Filters on Loopback Interface | **1835**

Configuring MPLS Firewall Filters and Policers on Switches | **1837**

- Configuring an MPLS Firewall Filter | **1837**
- Applying an MPLS Firewall Filter to an MPLS Interface | **1838**
- Applying an MPLS Firewall Filter to a Loopback Interface | **1838**
- Configuring Policers for LSPs | **1839**

Configuring MPLS Firewall Filters and Policers on Routers | **1840**

Configuring MPLS Firewall Filters and Policers | **1849**

Understanding How a Firewall Filter Tests a Protocol | **1852**

Understanding Firewall Filter Processing Points for Bridged and Routed Packets | **1853**

Understanding Filter-Based Forwarding | **1854**

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1855**

- Requirements | **1855**
- Overview and Topology | **1855**
- Configuration | **1856**
- Verification | **1859**

Configuring a Firewall Filter to De-Encapsulate GRE or IPIP Traffic | **1862**

- Configuring a Filter to De-Encapsulate GRE Traffic | **1862**
- Configuring a Filter to De-Encapsulate IPIP Traffic | **1864**
- Applying the Filter to an Interface | **1865**

Verifying That Firewall Filters Are Operational | **1866**

Monitoring Firewall Filter Traffic | **1867**

- Monitoring Traffic for All Firewall Filters and Policers That Are Configured on the Switch | **1867**
- Monitoring Traffic for a Specific Firewall Filter | **1869**
- Monitoring Traffic for a Specific Policer | **1869**

Troubleshooting Firewall Filter Configuration | **1871**

- Firewall Filter Configuration Returns a No Space Available in TCAM Message | **1871**
- Filter Counts Previously Dropped Packet | **1874**
- Matching Packets Not Counted | **1875**
- Counter Reset When Editing Filter | **1876**
- Cannot Include loss-priority and policer Actions in Same Term | **1876**

Cannot Egress Filter Certain Traffic Originating on QFX Switch | **1877**

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **1878**

Egress Firewall Filters with Private VLANs | **1878**

Egress Filtering of L2PT Traffic Not Supported | **1879**

Cannot Drop BGP Packets in Certain Circumstances | **1880**

Invalid Statistics for Policer | **1881**

Policers can Limit Egress Filters | **1881**

Configuring Firewall Filter Accounting and Logging (EX9200 Switches) | 1883

Example: Configuring Logging for a Stateless Firewall Filter Term | **1883**

Requirements | **1883**

Overview | **1883**

Configuration | **1884**

Verification | **1888**

Use the CLI Editor in Configuration Mode | **1889**

3 **Configuring Traffic Policers**

Understanding Traffic Policers | 1895

Policer Implementation Overview | **1896**

ARP Policer Overview | **1900**

Example: Configuring ARP Policer | **1902**

Requirements | **1902**

Overview | **1902**

Configuration | **1903**

Verification | **1905**

Understanding the Benefits of Policers and Token Bucket Algorithms | **1906**

Determining Proper Burst Size for Traffic Policers | **1909**

Controlling Network Access Using Traffic Policing Overview | **1916**

Traffic Policer Types | **1922**

Order of Policer and Firewall Filter Operations | **1926**

Understanding the Frame Length for Policing Packets | **1926**

Supported Standards for Policing | **1927**

Hierarchical Policer Configuration Overview | **1928**

Understanding Enhanced Hierarchical Policers | **1930**

Packets-Per-Second (pps)-Based Policer Overview | **1934**

Guidelines for Applying Traffic Policers | **1934**

Policer Support for Aggregated Ethernet Interfaces Overview | **1935**

Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface | **1937**

- Requirements | **1937**
- Overview | **1937**
- Configuration | **1938**
- Verification | **1944**

Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers | **1946**

Hierarchical Policers on ACX Series Routers Overview | **1949**

Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**

Hierarchical Policer Modes on ACX Series Routers | **1953**

Processing of Hierarchical Policers on ACX Series Routers | **1958**

Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**

Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Configuring Policer Rate Limits and Actions | 1966

Policer Bandwidth and Burst-Size Limits | **1966**

Policer Color-Marking and Actions | **1968**

Single Token Bucket Algorithm | **1971**

Dual Token Bucket Algorithms | **1974**

Configuring Layer 2 Policers | 1976

Hierarchical Policers | **1976**

- Hierarchical Policer Overview | **1977**
- Example: Configuring a Hierarchical Policer | **1978**
  - Requirements | **1979**
  - Overview | **1979**

Configuration | **1980**

Verification | **1986**

Example: Configuring a Hierarchical Policer for Subscriber Services Firewall (ACX7100-48L Devices) | **1987**

Overview | **1987**

Configuration Scenarios | **1988**

Configuring a Policer Overhead | **2001**

Two-Color and Three-Color Policers at Layer 2 | **2003**

Two-Color Policing at Layer 2 Overview | **2003**

Three-Color Policing at Layer 2 Overview | **2005**

Example: Configuring a Three-Color Logical Interface (Aggregate) Policer | **2008**

Requirements | **2008**

Overview | **2008**

Configuration | **2009**

Verification | **2015**

Layer 2 Traffic Policing at the Pseudowire Overview | **2017**

Configuring a Two-Color Layer 2 Policer for the Pseudowire | **2018**

Configuring a Three-Color Layer 2 Policer for the Pseudowire | **2019**

Applying the Policers to Dynamic Profile Interfaces | **2020**

Attaching Dynamic Profiles to Routing Instances | **2021**

Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview | **2023**

Configuring a Policer for the Complex Configuration | **2023**

Creating a Dynamic Profile for the Complex Configuration | **2024**

Attaching Dynamic Profiles to Routing Instances for the Complex Configuration | **2026**

Verifying Layer 2 Traffic Policers on VPLS Connections | **2027**

Understanding Policers on OVSDB-Managed Interfaces | **2028**

Example: Applying a Policer to OVSDB-Managed Interfaces | **2029**

Requirements | **2029**

Overview | **2030**

Configuration | **2030**

Configuring Two-Color and Three-Color Traffic Policers at Layer 3 | 2034

Two-Color Policer Configuration Overview | **2034**

Basic Single-Rate Two-Color Policers | **2042**

- Single-Rate Two-Color Policer Overview | **2042**
- Example: Limiting Inbound Traffic at Your Network Border by Configuring an Ingress Single-Rate Two-Color Policer | **2043**
  - Requirements | **2044**
  - Overview | **2044**
  - Configuration | **2047**
  - Verification | **2053**
- Example: Configuring Interface and Firewall Filter Policers at the Same Interface | **2055**
  - Requirements | **2055**
  - Overview | **2056**
  - Configuration | **2057**
  - Verification | **2067**

Bandwidth Policers | **2070**

- Bandwidth Policer Overview | **2070**
- Example: Configuring a Logical Bandwidth Policer | **2072**
  - Requirements | **2072**
  - Overview | **2073**
  - Configuration | **2074**
  - Verification | **2080**

Prefix-Specific Counting and Policing Actions | **2083**

- Prefix-Specific Counting and Policing Overview | **2084**
- Filter-Specific Counter and Policer Set Overview | **2087**
- Filter-Specific Policer Overview | **2087**
- Example: Configuring Prefix-Specific Counting and Policing | **2088**
  - Requirements | **2088**
  - Overview | **2088**
  - Configuration | **2090**
  - Verification | **2096**
- Prefix-Specific Counting and Policing Configuration Scenarios | **2098**

Policer Overhead to Account for Rate Shaping in the Traffic Manager | **2106**

- Policer Overhead to Account for Rate Shaping Overview | **2106**

Example: Configuring Policer Overhead to Account for Rate Shaping | **2107**

Requirements | **2107**

Overview | **2107**

Configuration | **2108**

Verification | **2116**

Three-Color Policer Configuration Overview | **2118**

Applying Policers | **2122**

Overview of Applying Policers | **2122**

Applying Aggregate Policers | **2123**

Applying Hierarchical Policers on Enhanced Intelligent Queuing PICs | **2126**

Configuring Hierarchical Policers | **2129**

Configuring a Single-Rate Two-Color Policer | **2130**

Configuring a Single-Rate Color-Blind Policer | **2131**

Configuring a Two-Rate Tricolor Marker Policer | **2132**

Three-Color Policer Configuration Guidelines | **2134**

Platforms Supported for Three-Color Policers | **2134**

Color Modes for Three-Color Policers | **2135**

Naming Conventions for Three-Color Policers | **2136**

Basic Single-Rate Three-Color Policers | **2138**

Single-Rate Three-Color Policer Overview | **2138**

Example: Configuring a Single-Rate Three-Color Policer | **2139**

Requirements | **2139**

Overview | **2139**

Configuration | **2140**

Verification | **2146**

Basic Two-Rate Three-Color Policers | **2147**

Two-Rate Three-Color Policer Overview | **2147**

Example: Configuring a Two-Rate Three-Color Policer | **2149**

Requirements | **2149**

Overview | **2149**

Configuration | **2150**

Verification | **2155**

Example: Configuring a Two-Rate Three-Color Policer | **2157**

Requirements | **2157**

Overview | **2157**

Configuration | **2158**

Verification | **2163**

Configuring Logical and Physical Interface Traffic Policers at Layer 3 | 2166

Two-Color and Three-Color Logical Interface Policers | **2166**

Logical Interface (Aggregate) Policer Overview | **2166**

Example: Configuring a Two-Color Logical Interface (Aggregate) Policer | **2167**

Requirements | **2168**

Overview | **2168**

Configuration | **2169**

Verification | **2174**

Example: Configuring a Three-Color Logical Interface (Aggregate) Policer | **2176**

Requirements | **2176**

Overview | **2176**

Configuration | **2177**

Verification | **2183**

Two-Color and Three-Color Physical Interface Policers | **2185**

Physical Interface Policer Overview | **2185**

Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface | **2187**

Requirements | **2187**

Overview | **2187**

Configuration | **2188**

Verification | **2194**

Configuring Policers on Switches | 2197

Overview of Policers | **2198**

Traffic Policer Types | **2206**

Understanding the Use of Policers in Firewall Filters | **2210**

Understanding Tricolor Marking Architecture | **2215**

Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**

Configuring Policers | **2216**

Specifying Policers in a Firewall Filter Configuration | **2218**

Applying a Firewall Filter That Is Configured with a Policer | **2218**

Configuring Tricolor Marking Policers | **2218**

Configuring a Tricolor Marking Policer | **2219**

Applying Tricolor Marking Policers to Firewall Filters | **2220**

Understanding Policers with Link Aggregation Groups | **2221**

Understanding Color-Blind Mode for Single-Rate Tricolor Marking | **2222**

Understanding Color-Aware Mode for Single-Rate Tricolor Marking | **2222**

Understanding Color-Blind Mode for Two-Rate Tricolor Marking | **2225**

Understanding Color-Aware Mode for Two-Rate Tricolor Marking | **2225**

Example: Using Two-Color Policers and Prefix Lists | **2228**

Example: Using Policers to Manage Oversubscription | **2232**

Assigning Forwarding Classes and Loss Priority | **2234**

Configuring Color-Blind Egress Policers for Medium-Low PLP | **2237**

Configuring Two-Color and Three-Color Policers to Control Traffic Rates | **2237**

Configuring Two-Color Policers | **2238**

Configuring Three-Color Policers | **2239**

Specifying Policers in a Firewall Filter Configuration | **2240**

Applying a Firewall Filter That Includes a Policer | **2240**

Verifying That Two-Color Policers Are Operational | **2241**

Verifying That Three-Color Policers Are Operational | **2242**

Troubleshooting Policer Configuration | **2243**

Incomplete Count of Packet Drops | **2243**

Counter Reset When Editing Filter | **2244**

Invalid Statistics for Policer | **2245**

Policers Can Limit Egress Filters | **2245**

Troubleshooting Policer Configuration | **2247**

Incomplete Count of Packet Drops | **2247**

Counter Reset When Editing Filter | **2248**

Invalid Statistics for Policer | **2248**

Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured | **2249**

Filter-Specific Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured | **2250**

Policers Can Limit Egress Filters | **2251**

4 **Configuration Statements and Operational Commands**

Firewall Filter Configuration Statements Supported by Junos OS for EX Series Switches | 2254

gtp-header | 2259

gtp-teid | 2261

per-logical-interface-firewall | 2263

Junos CLI Reference Overview | 2265

5 **Troubleshooting**

Knowledge Base | 2267

About This Guide

Routing policies allow you to control the routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. All routing protocols use the Junos OS routing tables to store the routes that they learn and to determine which routes they should advertise in their protocol packets. Routing policies also allow you to control which routes the routing protocols store in and retrieve from the routing table.

Firewall filter policies allow you to control packets transiting the router to a network destination and packets destined for and sent by the router. They 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.

RELATED DOCUMENTATION

- Day One: Configuring Junos Policies and Firewall Filters

PART 1

Understanding and Configuring Junos Routing Policies

Overview | 2

Evaluating Routing Policies Using Match Conditions, Actions, Terms, and Expressions | 54

Evaluating Complex Cases Using Policy Chains and Subroutines | 256

Configuring Route Filters and Prefix Lists as Match Conditions | 303

Configuring AS Paths as Match Conditions | 433

Configuring Communities as Match Conditions | 511

Increasing Network Stability with BGP Route Flapping Actions | 596

Tracking Traffic Usage with Source Class Usage and Destination Class Usage Actions | 635

Avoiding Traffic Routing Threats with Conditional Routing Policies | 686

Protecting Against DoS Attacks by Forwarding Traffic to the Discard Interface | 714

Improving Commit Times with Dynamic Routing Policies | 732

Testing Before Applying Routing Policies | 758

CHAPTER 1

Overview

IN THIS CHAPTER

- Policy Framework Overview | 2
- Comparison of Routing Policies and Firewall Filters | 9
- Prefix Prioritization Overview | 15
- FIB Prefix Prioritization | 16
- Accounting of the Policer Overhead Attribute at the Interface Level | 17
- Configuring the Accounting of Policer Overhead in Interface Statistics | 19
- Understanding Routing Policies | 22
- Protocol Support for Import and Export Policies | 26
- Example: Applying Routing Policies at Different Levels of the BGP Hierarchy | 27
- Default Routing Policies | 40
- Example: Configuring a Conditional Default Route Policy | 44

Policy Framework Overview

IN THIS SECTION

- Routing Policy and Firewall Filters | 3
- Reasons to Create a Routing Policy | 3
- Router Flows Affected by Policies | 4
- Control Points | 7
- Policy Components | 8

The Junos® operating system (Junos OS) provides a *policy framework*, which is a collection of Junos OS policies that allows you to control flows of routing information and packets.

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

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

Routing Policy and Firewall Filters

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. All routing protocols use the Junos OS routing tables to store the routes that they learn and to determine which routes they should advertise in their protocol packets. Routing policy allows you to control which routes the routing protocols store in and retrieve from the routing 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.

Reasons to Create a Routing Policy

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).

Router Flows Affected by Policies

The Junos OS policies affect the following router flows:

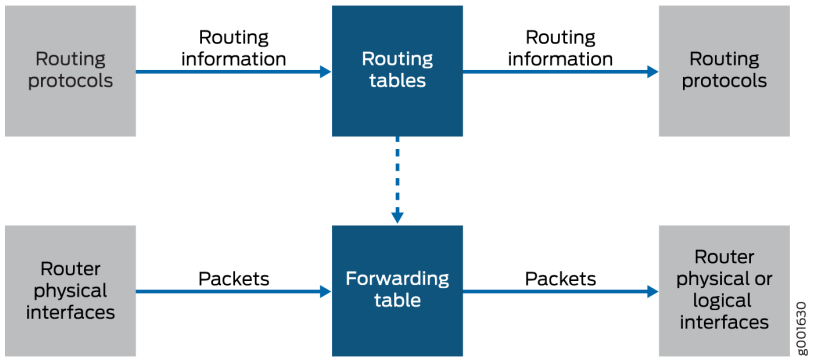
- Flow of routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. The Routing Engine handles this flow. *Routing information* is the information about routes learned by the routing protocols from a router's neighbors. This information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

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

Figure 1 on page 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

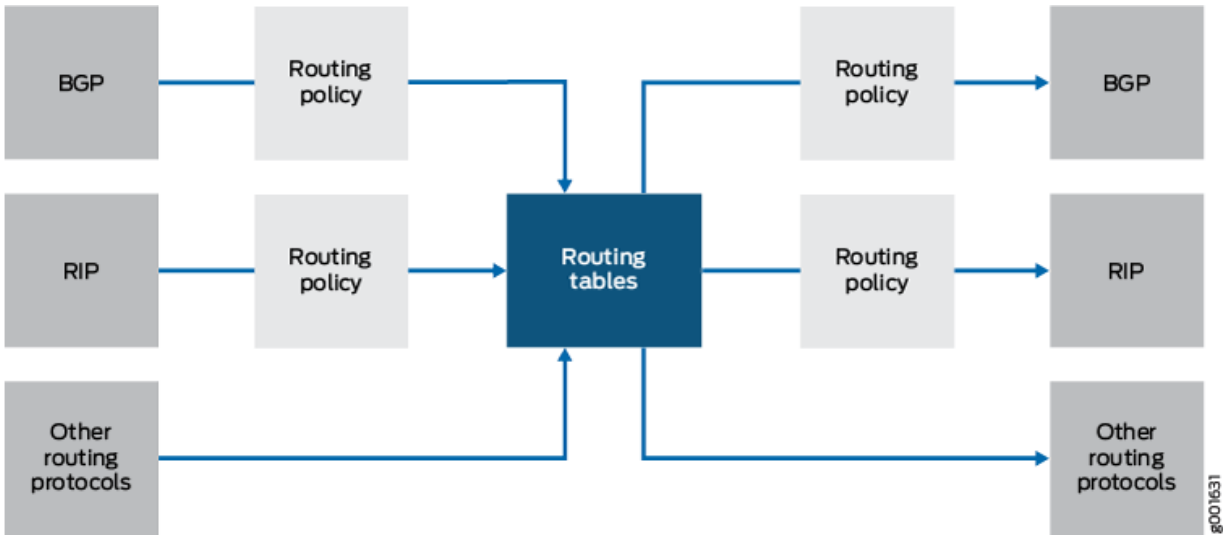


You can configure routing policies to control which routes the routing protocols place in the routing tables and to control which routes the routing protocols advertise from the routing tables (see Figure 2 on page 6). The routing protocols advertise active routes only from the routing tables. (An *active route* is a route that is chosen from all routes in the routing table to reach a destination.)

You can also use routing policies to do the following:

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

Figure 2: Routing Policies to Control Routing Information Flow

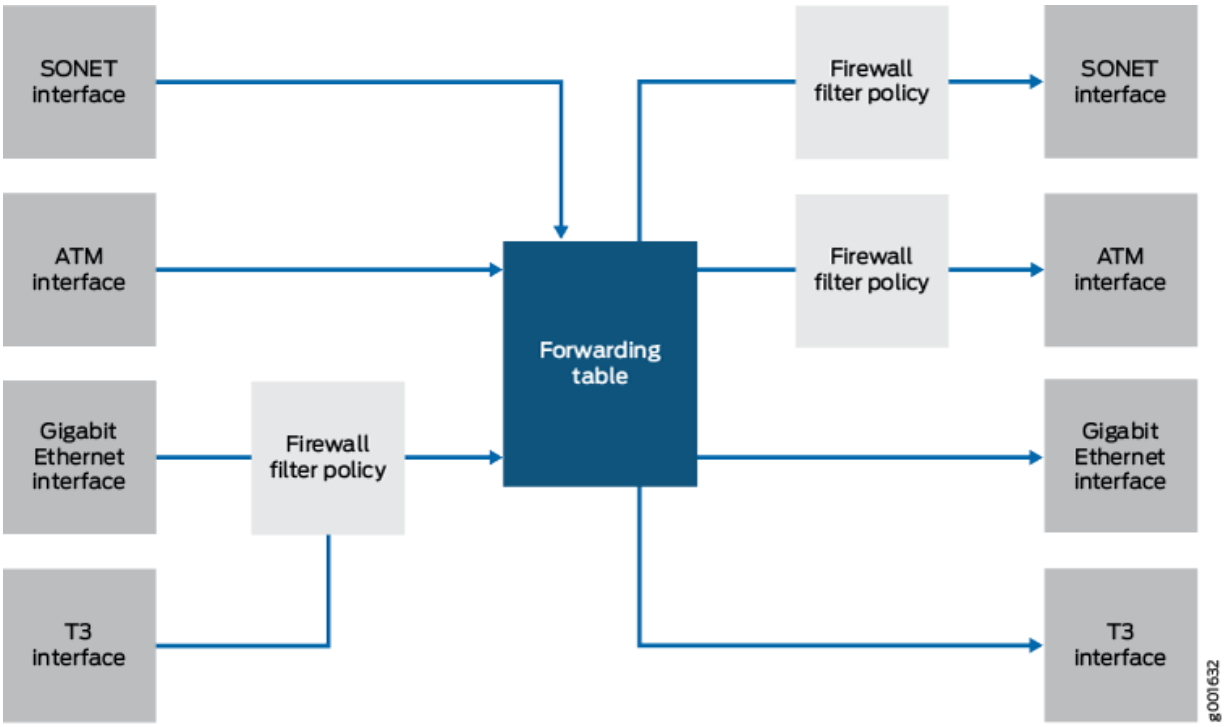


You can configure firewall filters to control the following aspects of packet flow (see Figure 3 on page 7):

- 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

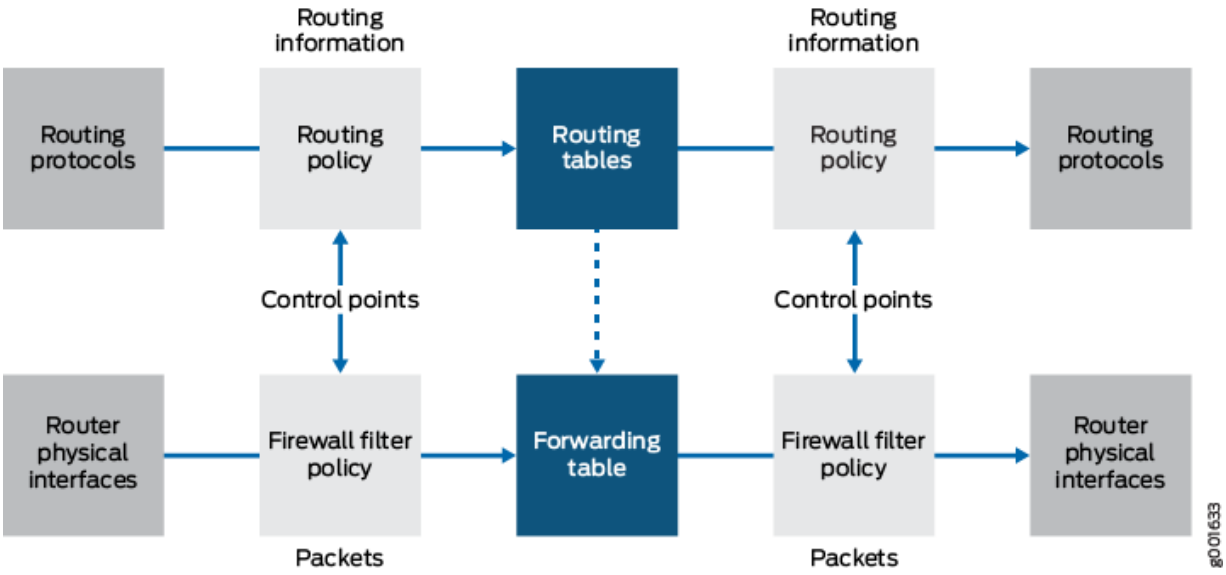


Control Points

All policies provide two points at which you can control routing information or packets through the router (see Figure 4 on page 8). 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 8 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.

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. 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.

RELATED DOCUMENTATION

Comparison of Routing Policies and Firewall Filters | 9

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Comparison of Routing Policies and Firewall Filters

Although routing policies and firewall filters share an architecture, their purposes, implementation, and configuration are different. Table 1 on page 9 describes their purposes. Table 2 on page 10 compares the implementation details for routing policies and firewall filters, highlighting the similarities and differences in their configuration.

Table 1: 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 2: Implementation Differences Between Routing Policies and Firewall Filters

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Control points	Control routing information that is placed in the routing table with an import routing policy and advertised from the routing table with an export routing policy.	Control packets that are accepted on a router interface with an input firewall filter and that are forwarded from an interface with an output firewall filter.
Configuration tasks: • Define policy • Apply policy	Define a policy that contains terms, match conditions, and actions. Apply one or more export or import policies to a routing protocol. You can also apply a *policy expression*, which uses Boolean logical operators with multiple import or export policies. You can also apply one or more export policies to the forwarding table.	Define a policy that contains terms, match conditions, and actions. Apply one input or output firewall filter to a physical interface or physical interface group to filter data packets received by or forwarded to a physical interface (on routing platforms with an Internet Processor II application-specific integrated circuit [ASIC] only). You can also apply one input or output firewall filter to the routing platform's loopback interface, which is the interface to the Routing Engine (on all routing platforms). This allows you to filter local packets received by or forwarded from the Routing Engine.
Terms	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a policy ends after a packet matches the criteria in a term and the defined or default policy action of accept or reject is taken. The route is not evaluated against subsequent terms in the same policy or subsequent policies.	Configure as many terms as desired. Define a name for each term. Terms are evaluated in the order in which you specify them. Evaluation of a firewall filter ends after a packet matches the criteria in a term and the defined or default action is taken. The packet is not evaluated against subsequent terms in the firewall filter.

Table 2: Implementation Differences Between Routing Policies and Firewall Filters *(Continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Match conditions	Specify zero or more criteria that a route must match. You can specify criteria based on source, destination, or properties of a route. You can also specify the following match conditions, which require more configuration: • Autonomous system (AS) path expression—A combination of AS numbers and regular expression operators. • Community—A group of destinations that share a common property. • Prefix list—A named list of prefixes. • Route list—A list of destination prefixes. • Subroutine—A routing policy that is called repeatedly from other routing policies.	Specify zero or more criteria that a packet must match. You must match various fields in the packet's header. The fields are grouped into the following categories: • Numeric values, such as port and protocol numbers. • Prefix values, such as IP source and destination prefixes. • Bit-field values—Whether particular bits in the fields are set, such as IP options, Transmission Control Protocol (TCP) flags, and IP fragmentation fields. You can specify the fields using Boolean logical operators.

Table 2: Implementation Differences Between Routing Policies and Firewall Filters *(Continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
Actions	Specify zero or one action to take if a route matches all criteria. You can specify the following actions: • Accept—Accept the route into the routing table, and propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. • Reject—Do not accept the route into the routing table, and do not propagate it. After this action is taken, the evaluation of subsequent terms and policies ends. In addition to the preceding actions, you can also specify zero or more of the following types of actions: • Next term—Evaluate the next term in the routing policy. • Next policy—Evaluate the next routing policy. • Actions that manipulate characteristics associated with a route as the routing protocol places it in the routing table or advertises it from the routing table. • Trace action, which logs route matches.	Specify zero or one action to take if a packet matches all criteria. (We recommend that you always explicitly configure an action.) You can specify the following actions: • Accept—Accept a packet. • Discard—Discard a packet silently, without sending an ICMP message. • Reject—Discard a packet, and send an ICMP destination unreachable message. • Routing instance—Specify a routing table to which packets are forwarded. • Next term—Evaluate the next term in the firewall filter. **NOTE**: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported. In addition to zero or the preceding actions, you can also specify zero or more action modifiers. You can specify the following action modifiers: • Count—Add packet to a count total. • Forwarding class—Set the packet forwarding class to a specified value from 0 through 3. • IPsec security association—Used with the source and destination address match conditions, specify an IP Security (IPsec) security association (SA) for the packet.

Table 2: Implementation Differences Between Routing Policies and Firewall Filters *(Continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation
		• 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 2: Implementation Differences Between Routing Policies and Firewall Filters *(Continued)*

Policy Architecture	Routing Policy Implementation	Firewall Filter Implementation

Default policies and actions	If an incoming or outgoing route arrives and a policy related to the route is not explicitly configured, the action specified by the default policy for the associated routing protocol is taken.
	
	The following default actions exist for routing policies:
	
	• If a policy does not specify a match condition, all routes evaluated against the policy match.
	
	• If a match occurs but the policy does not specify an accept, reject, next term, or next policy action, one of the following occurs:
	
	  • The next term, if present, is evaluated.
	
	  • If no other terms are present, the next policy is evaluated.
	
	  • If no other policies are present, the action specified by the default policy is taken.
	
	• If a match does not occur with a term in a policy and subsequent terms in the same policy exist, the next term is evaluated.
	
	• If a match does not occur with any terms in a policy and subsequent policies exist, the next policy is evaluated.
	
	• If a match does not occur by the end of a policy and no other policies exist, the accept or reject action specified by the default policy is taken.
		If an incoming or outgoing packet arrives on an interface and a firewall filter is not configured for the interface, the default policy is taken (the packet is accepted).
		
		The following default actions exist for firewall filters:
		
		• If a firewall filter does not specify a match condition, all packets are considered to match.
		
		• If a match occurs but the firewall filter does not specify an action, the packet is accepted.
		
		• If a match occurs, the defined or default action is taken and the evaluation ends. Subsequent terms in the firewall filter are not evaluated, unless the `next term` action is specified.
		
		NOTE: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.
		
		• 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.

RELATED DOCUMENTATION

Policy Framework Overview | 2

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Understanding the Algorithm Used to Load Balance Traffic on MX Series Routers

Prefix Prioritization Overview

Junos OS routes have a predetermined order for route installation. This is no longer sufficient as it is sometimes required to prioritize certain routes or prefixes over others for better convergence or to provide differentiated services. In a network with a large number of routes, it is sometimes important to control the order in which routes get updated due to a change in the network topology. For example, it would be useful to first update OSPF routes that correspond to an IBGP neighbor, and only then update the rest of the OSPF routes. At a system level, Junos OS implements reasonable defaults based on heuristics to determine the order in which routes get updated. However, the default behavior is not always optimal. Prefix prioritization allows the user to control the order in which the routes get updated from LDP or OSPF to rpd, and from rpd to kernel. In Junos OS Release 16.1 and later, you can control the order in which the routes get updated from LDP or OSPF to rpd, and from rpd to kernel.

In Junos OS Release 16.1 and later, the Junos OS policy language is extended to let the user set the relative priority `high` and `low` for prefixes via the existing protocol import policy. Based on the tagged priority, the routes are placed in different priority queues. Routes that are not explicitly assigned a priority are treated as medium priority. Within the same priority level, routes will continue to be updated in lexicographic order. Prefix prioritization would need each supporting protocol to prioritize its routes internally. Prefix prioritization ensures that high priority IGP and LDP routes get updated to the FIB (forwarding table) before medium and low priority routes.

NOTE: There is an upper limit on how many high priority prefixes are allowed in the kernel. Not more than 10,000 high priority prefixes can coexist in the kernel. If this threshold is crossed in the kernel, then any new high priority prefix addition request will be considered as normal priority. This is a “best effort” prioritization scheme and will not be handled if the kernel is highly loaded.

RELATED DOCUMENTATION

Example: Configuring the Priority for Route Prefixes in the RPD Infrastructure | 412

Configuring Priority for Route Prefixes in RPD Infrastructure | 426

FIB Prefix Prioritization

IN THIS SECTION

- FIB prefix prioritization | 16
- FIB prefix priorities | 16
- Supported route types for FIB prefix prioritization in order of install preference | 16
- FIB prefix prioritization workflow | 17

FIB prefix prioritization

FIB prefix prioritization allows user-defined priorities to be assigned to routes when they are exported to the forwarding plane from the routing plane. Route priorities can be assigned in the routing plane, by way of IGP protocol import policies that assign priorities to routes. The user sets the relative priority high and low for prefixes via the existing protocol import policy – see Prefix Prioritization Overview. These route priorities are exported into the forwarding table.

However, there could be situations when there is a need to override the routing plane route prioritization, with user-defined route prioritization at the forwarding plane. FIB (forwarding information base) prefix prioritization allows this. In the forwarding table export policy, on matching routes, a route priority can be assigned.

FIB prefix priorities

- High – Prefixes assigned this priority have the highest install priority. These routes are always given importance over others.
- Medium – Prefixes assigned this priority have the second highest install priority.

NOTE: Prefixes that are not assigned high or medium priorities are unprioritized.

Supported route types for FIB prefix prioritization in order of install preference

- Interface/local routes – Given highest priority unconditionally and will bypass forwarding table export policy evaluation.
- Host routes
- IPv4 and IPv6 routes
- MPLS – routes protization will be PROTO-based and not prefix-based.

FIB prefix prioritization workflow

The forward table reserves memory buffers for high and medium priority routes, based on user configuration – see *fib-prioritization*. Once this configuration is set, the number of lower priority routes that can be installed in the forwarding table is limited to the remaining space for these routes.

After setting the percentages for high and medium priority routes, the next step is to configure the forwarding table export policy. See *fib-install-priority*.

Once routes are installed in the forwarding table and have their route priorities assigned by the FIB prefix prioritization process, the routes can transition to other priorities as well. Because iterations of a forwarding table export policy might mark a previously unprioritized route as high priority for example, such transitions are managed in the forwarding table by the packet forwarding engine.

Accounting of the Policer Overhead Attribute at the Interface Level

IN THIS SECTION

- Need for Policer Overhead adjustment | **18**
- Policer Overhead Adjustment Applicability for Policers | **18**

A bandwidth profile (BWP) enforces limits on bandwidth utilization according to the service level specification (SLS) that has been agreed upon by the subscriber and the service provider as part of the service level agreement (SLA).There are two types of bandwidth profiles:

- Ingress Bandwidth Profile
- Egress Bandwidth Profile

Need for Policer Overhead adjustment

The Metro Ethernet Forum (MEF) Carrier Ethernet (CE) 2.0 set of standards has stringent requirements for the bandwidth profile enforcement at the user network interface (UNI). MEF CE 2.0 certification compliance allows only a 2 percent deviation from UNI commited or peak rate across all frame sizes. This means that the policers should take into account the frame size at the UNI interface, including frame checksum and excluding all additional overheads that might be added inside the service provider network (such as S-VLANs). Therefore, this translates into two customer requirements:

- Junos OS egress policers use frame length before output VLAN manipulation. If VLANs are added on output, those extra bytes will not be counted. In order to address MEF CE 2.0 requirements, adjust the length of the packet that is used for policing purposes for Junos egress policers that use frame length before output VLAN manipulation. Therefore, if VLANs are added on output, the extra bytes will not be counted.

- In some network designs, bandwidth profile enforcement is implemented at the Layer 2 (L2) VPN Provider Edge Router and not at the Ethernet access device (EAD) terminating the physical UNI interface. The EAD typically adds an S-VLAN that identifies the port in the access network. The S-VLAN that is added is considered internal to the service provider network and typically should not be taken into account for bandwidth profile enforcement purposes at the Provider Edge device in both ingress and egress directions. This also translates into a requirement to allow adjusting the packet length used for policing purposes on ingress and egress.

In order to address these requirements, `policer-overhead` adjustment is defined on a per logical interface (IFL)/direction granularity, which is the range of -16 bytes to +16 bytes. The `policer-overhead` adjustment is applied for all the policers that take into account Layer 1 (L1) or L2 packet length that are exercised in the specified IFL/direction, including corresponding logical interface family (IFF) feature policers, and is applied only to interface or filter-based policers.

Policer Overhead Adjustment Applicability for Policers

The ingress or egress `policer overhead` adjustment configuration is applicable for the following types of policers on ingress or egress IFL and IFF, respectively:

- L2 two-color and three-color policers.

- IFL-level policers (configured at the IFL or in a filter attached to IFL).

- Family-level policers that use L2 packet length, or policers in filters attached to L2 IFF (family bridge, vpls, ccc).

NOTE: The list is applicable for all types of policers, including regular two-color policers, three-color policers, and hierarchical policers, provided that the policer operates on an L1 or L2 packet length.

`Ingress policer overhead adjustment` configuration is applied to any policers attached to ingress L2 routing instances.

NOTE: Note that any IFF policer on the L3 family (inet inet6), which considers only L3 packet length, will not consider this adjustment. The `policer overhead` adjustment value (+ve or -ve) is added to the actual L2 packet length to obtain the number of bytes to charge the policer. Therefore, this is used only as an intermediate value, and does not affect actual L2 packet length. This feature is applied before VLAN normalization, and is independent of the `egress-shaping-overhead` or `ingress-shaping-overhead` configuration under class of service.

RELATED DOCUMENTATION

policer-overhead-adjustment

Configuring the Accounting of Policer Overhead in Interface Statistics | 19

Configuring the Accounting of Policer Overhead in Interface Statistics

The Metro Ethernet Forum (MEF) Carrier Ethernet (CE) 2.0 set of standards has stringent requirements to the bandwidth profile enforcement at the user-to-network interfaces (UNI). MEF CE 2.0 certification compliance allows only a 2 percent deviation from UNI committed or peak rate across all frame sizes. This means that the policers should take into account the frame size at the UNI interface, including frame checksum and excluding all additional overheads that might be added inside the service provider network (such as S-VLANs).

In order to address the MEF CE 2.0 stringent requirements to the bandwidth profile, `policer-overhead` adjustment is defined on a per IFL or direction granularity. The `policer-overhead` adjustment is in the range of -16 bytes to +16 bytes and is applied for all the policers that take into account Layer 1/ Layer 2 (L1/L2) packet length in the specified IFL or direction, including corresponding logical interface family (IFF) feature policers.

To configure the policer-overhead:

1. At the [edit interfaces] hierarchy level in configuration mode, create the interface on which to add the policer-overhead to input or output traffic.

```
[edit interfaces]
user@host# edit interfaces interface name
```

For example:

```
[edit interfaces interface name]
user@host# edit xe-0/1/6
```

2. Create the unit on which to add the policer overhead.

```
[edit interfaces unit]
[edit interfaces interface name unit unit-number]
```

For example:

```
[edit interfaces unit]
user@host# edit xe-0/1/6 unit 0
```

3. Configure the policer-overhead for the ingress policer.

```
[edit interfaces interface name unit unit-number ]
user@host# set policer-overhead ingress value in bytes (-16..+16 bytes)
```

For example:

```
[edit xe-0/1/6 unit 0]
user@host# set policer-overhead ingress 10;
```

4. Configure the policer-overhead for the egress policer.

```
user@host# set policer-overhead egress value in bytes (-16..+16 bytes)
```

```
[edit xe-0/1/6 unit 0]
user@host# set policer-overhead egress 10;
```

5. Verify the configuration.

```
[edit interfaces]
user@host# show interfaces xe-0/1/6
xe-0/1/6 {
        unit 0 {
            policer-overhead {
                ingress 10;
                egress 10;
            }
        }
    }
user@host>show interfaces xe-0/1/6
Physical interface: xe-0/1/6, Enabled, Physical link is Up
  Interface index: 161, SNMP ifIndex: 544
  Link-level type: Ethernet, MTU: 1514, MRU: 1522, LAN-PHY mode, Speed: 10Gbps, BPDU Error:
None, MAC-REWRITE Error: None,
  Loopback: None, Source filtering: Disabled, Flow control: Enabled
  Pad to minimum frame size: Disabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Current address: 00:23:9c:fc:a8:58, Hardware address: 00:23:9c:fc:a8:58
  Last flapped   : 2015-09-13 20:12:36 PDT (14:21:57 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Link Degrade :
    Link Monitoring                    :  Disable
  Interface transmit statistics: Disabled

  Logical interface xe-0/1/6.0 (Index 339) (SNMP ifIndex 571)
```

```
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Policer overhead :
     Ingress policer overhead  :                    10 bytes
     Egress policer overhead   :                    10 bytes
    Input packets : 0
    Output packets: 0
    Protocol multiservice, MTU: Unlimited
      Flags: Is-Primary



user@host> show interfaces xe-0/1/6.0
  Logical interface xe-0/1/6.0 (Index 339) (SNMP ifIndex 571)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Policer overhead :
     Ingress policer overhead  :                    10 bytes
     Egress policer overhead   :                    10 bytes
    Input packets : 0
    Output packets: 0
    Protocol multiservice, MTU: Unlimited
      Flags: Is-Primary
```

RELATED DOCUMENTATION

policer-overhead-adjustment

Accounting of the Policer Overhead Attribute at the Interface Level | 17

Understanding Routing Policies

IN THIS SECTION

- Importing and Exporting Routes | 23
- Active and Inactive Routes | 25
- Explicitly Configured Routes | 25
- Dynamic Database | 25

For some routing platform vendors, the flow of routes occurs between various protocols. If, for example, you want to configure redistribution from RIP to OSPF, the RIP process tells the OSPF process that it has routes that might be included for redistribution. In Junos OS, there is not much direct interaction between the routing protocols. Instead, there are central gathering points where all protocols install their routing information. These are the main unicast routing tables inet.0 and inet6.0.

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.

Importing and Exporting Routes

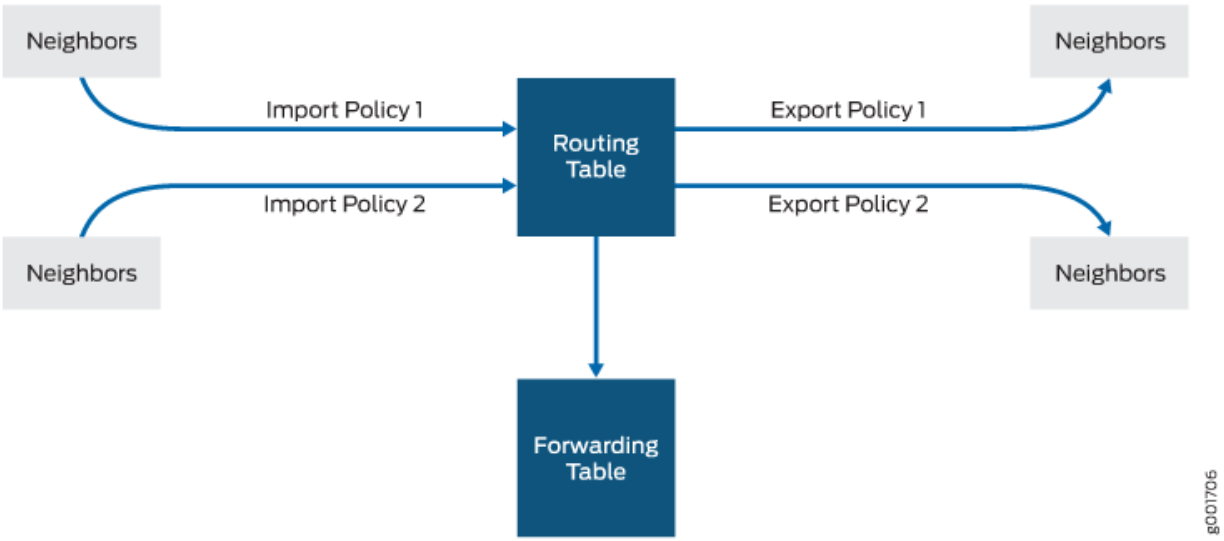
Two terms—*import* and *export*—explain how routes move between the routing protocols and the routing table.

- 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.

As shown in Figure 5 on page 24, you use import routing policies to control which routes are placed in the routing table, and export routing policies to control which routes are advertised from the routing table to neighbors.

Figure 5: Importing and Exporting Routes



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* allows you to modify the routing policy framework to suit your needs. You can create and implement your own routing policies to do the following:

- Control which routes a routing protocol places in the routing table.
- Control which active routes a routing protocol advertises from the routing table. An *active route* is a route that is chosen from all routes in the routing table to reach a destination.
- Manipulate the route characteristics as a routing protocol places the route in the routing table or advertises the route 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 is used to forward traffic toward the route's destination. In general, the active route is also advertised to a router's neighbors.

Active and Inactive Routes

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

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

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

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

Explicitly Configured Routes

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.

Dynamic Database

In Junos OS Release 9.5 and later, you can configure routing policies and certain routing policy objects in a dynamic database that is not subject to the same verification required by the standard configuration database. As a result, you can quickly commit these routing policies and policy objects, which can be referenced and applied in the standard configuration as needed. BGP is the only protocol to which you can apply routing policies that reference policies configured in the dynamic database. After a routing policy based on the dynamic database is configured and committed in the standard configuration, you can quickly make changes to existing routing policies by modifying policy objects in the dynamic

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

RELATED DOCUMENTATION

Example: Configuring Dynamic Routing Policies

Protocol Support for Import and Export Policies

Table 3: 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	Yes	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

Table 3: Protocol Support for Import and Export Policies *(Continued)*

Protocol	Import Policy	Export Policy	Supported Levels
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

Example: Applying Routing Policies at Different Levels of the BGP Hierarchy

IN THIS SECTION

- Requirements | 27
- Overview | 28
- Configuration | 30
- Verification | 36

This example shows BGP configured in a simple network topology and explains how routing polices take effect when they are applied at different levels of the BGP configuration.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 29

For BGP, you can apply policies as follows:

- BGP global `import` and `export` statements—Include these statements at the `[edit protocols bgp]` hierarchy level (for routing instances, include these statements at the `[edit routing-instances` *`routing-instance-name`* `protocols bgp]` hierarchy level).
- Group `import` and `export` statements—Include these statements at the `[edit protocols bgp group` *`group-name`*`]` hierarchy level (for routing instances, include these statements at the `[edit routing-instances` *`routing-instance-name`* `protocols bgp group` *`group-name`*`]` hierarchy level).
- Peer `import` and `export` statements—Include these statements at the `[edit protocols bgp group` *`group-name`* `neighbor` *`address`*`]` hierarchy level (for routing instances, include these statements at the `[edit routing-instances` *`routing-instance-name`* `protocols bgp group` *`group-name`* `neighbor` *`address`*`]` hierarchy level).

A peer-level `import` or `export` statement overrides a group `import` or `export` statement. A group-level `import` or `export` statement overrides a global BGP `import` or `export` statement.

In this example, a policy named `send-direct` is applied at the global level, another policy named `send-192.168.0.1` is applied at the group level, and a third policy named `send-192.168.20.1` is applied at the neighbor level.

```
user@host# show protocols
bgp {
    local-address 172.16.1.1;
    export send-direct;
    group internal-peers {
        type internal;
        export send-192.168.0.1;
        neighbor 172.16.2.2 {
            export send-192.168.20.1;
        }
        neighbor 172.16.3.3;
    }
    group other-group {
```

```
        type internal;
        neighbor 172.16.4.4;
    }
}
```

A key point, and one that is often misunderstood and that can lead to problems, is that in such a configuration, only the most explicit policy is applied. A neighbor-level policy is more explicit than a group-level policy, which in turn is more explicit than a global policy.

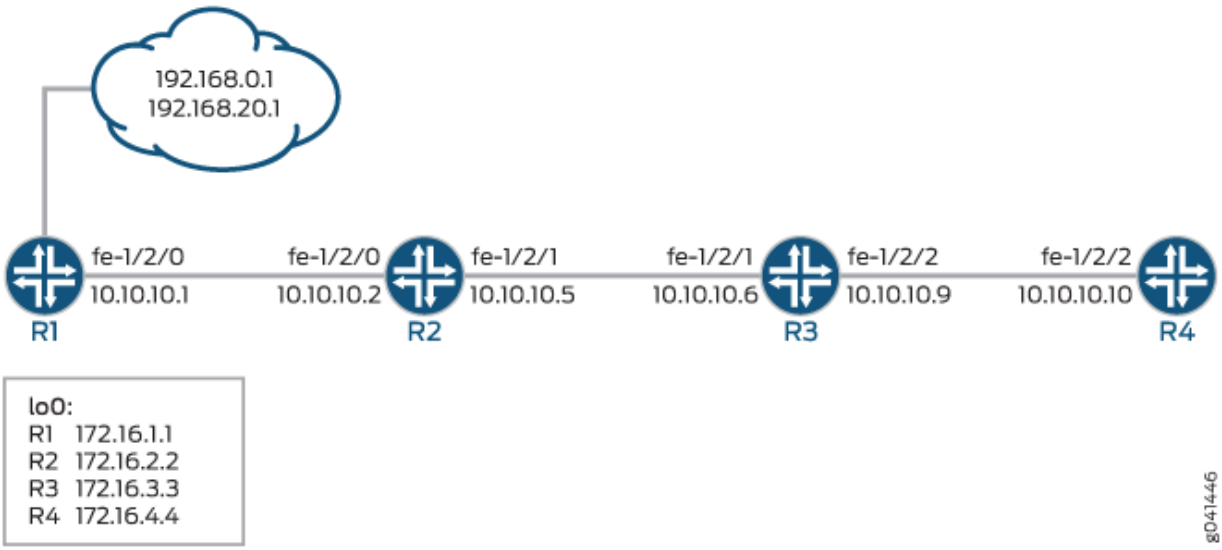
The neighbor 172.16.2.2 is subjected only to the send-192.168.20.1 policy. The neighbor 172.16.3.3, lacking anything more specific, is subjected only to the send-192.168.0.1 policy. Meanwhile, neighbor 172.16.4.4 in group other-group has no group or neighbor-level policy, so it uses the send-direct policy.

If you need to have neighbor 172.16.2.2 perform the function of all three policies, you can write and apply a new neighbor-level policy that encompasses the functions of the other three, or you can apply all three existing policies, as a chain, to neighbor 172.16.2.2.

Topology

Figure 6 on page 29 shows the sample network.

Figure 6: Applying Routing Policies to BGP



"CLI Quick Configuration" on page 30 shows the configuration for all of the devices in Figure 6 on page 29.

The section "No Link Title" on page 32 describes the steps on Device R1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 30
- Procedure | 32
- Results | 34

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 description to-R2
set interfaces fe-1/2/0 unit 0 family inet address 10.10.10.1/30
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set protocols bgp local-address 172.16.1.1
set protocols bgp export send-direct
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers export send-static-192.168.0
set protocols bgp group internal-peers neighbor 172.16.2.2 export send-static-192.168.20
set protocols bgp group internal-peers neighbor 172.16.3.3
set protocols bgp group other-group type internal
set protocols bgp group other-group neighbor 172.16.4.4
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement send-static-192.168.0 term 1 from protocol static
set policy-options policy-statement send-static-192.168.0 term 1 from route-filter
192.168.0.0/24 orlonger
set policy-options policy-statement send-static-192.168.0 term 1 then accept
set policy-options policy-statement send-static-192.168.20 term 1 from protocol static
set policy-options policy-statement send-static-192.168.20 term 1 from route-filter
192.168.20.0/24 orlonger
set policy-options policy-statement send-static-192.168.20 term 1 then accept
```

```
set routing-options static route 192.168.0.1/32 discard
set routing-options static route 192.168.20.1/32 discard
set routing-options router-id 172.16.1.1
set routing-options autonomous-system 17
```

Device R2

```
set interfaces fe-1/2/0 unit 0 description to-R1
set interfaces fe-1/2/0 unit 0 family inet address 10.10.10.2/30
set interfaces fe-1/2/1 unit 0 description to-R3
set interfaces fe-1/2/1 unit 0 family inet address 10.10.10.5/30
set interfaces lo0 unit 0 family inet address 172.16.2.2/32
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers local-address 172.16.2.2
set protocols bgp group internal-peers neighbor 172.16.3.3
set protocols bgp group internal-peers neighbor 172.16.1.1
set protocols bgp group internal-peers neighbor 172.16.4.4
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set routing-options router-id 172.16.2.2
set routing-options autonomous-system 17
```

Device R3

```
set interfaces fe-1/2/1 unit 0 description to-R2
set interfaces fe-1/2/1 unit 0 family inet address 10.10.10.6/30
set interfaces fe-1/2/2 unit 0 description to-R4
set interfaces fe-1/2/2 unit 0 family inet address 10.10.10.9/30
set interfaces lo0 unit 0 family inet address 172.16.3.3/32
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers local-address 172.16.3.3
set protocols bgp group internal-peers neighbor 172.16.2.2
set protocols bgp group internal-peers neighbor 172.16.1.1
set protocols bgp group internal-peers neighbor 172.16.4.4
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set routing-options router-id 172.16.3.3
set routing-options autonomous-system 17
```

Device R4

```
set interfaces fe-1/2/2 unit 0 description to-R3
set interfaces fe-1/2/2 unit 0 family inet address 10.10.10.10/30
set interfaces lo0 unit 0 family inet address 172.16.4.4/32
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers local-address 172.16.4.4
set protocols bgp group internal-peers neighbor 172.16.2.2
set protocols bgp group internal-peers neighbor 172.16.1.1
set protocols bgp group internal-peers neighbor 172.16.3.3
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set routing-options router-id 172.16.4.4
set routing-options autonomous-system 17
```

Procedure

Step-by-Step Procedure

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

To configure an IS-IS default route policy:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 0 description to-R2
user@R1# set fe-1/2/0 unit 0 family inet address 10.10.10.1/30
user@R1# set lo0 unit 0 family inet address 172.16.1.1/32
```

2. Enable OSPF, or another interior gateway protocols (IGP), on the interfaces.

```
[edit protocols OSPF area 0.0.0.0]
user@R1# set interface lo0.0 passive
user@R1# set interface fe-1/2/0.0
```

3. Configure static routes.

```
[edit routing-options]
user@R1# set static route 192.168.0.1/32 discard
user@R1# set static route 192.168.20.1/32 discard
```

4. Enable the routing policies.

```
[edit protocols policy-options]
user@R1# set policy-statement send-direct term 1 from protocol direct
user@R1# set policy-statement send-direct term 1 then accept
user@R1# set policy-statement send-static-192.168.0 term 1 from protocol static
user@R1# set policy-statement send-static-192.168.0 term 1 from route-filter 192.168.0.0/24
orlonger
user@R1# set policy-statement send-static-192.168.0 term 1 then accept
user@R1# set policy-statement send-static-192.168.20 term 1 from protocol static
user@R1# set policy-statement send-static-192.168.20 term 1 from route-filter 192.168.20.0/24
orlonger
user@R1# set policy-statement send-static-192.168.20 term 1 then accept
```

5. Configure BGP and apply the export policies.

```
[edit protocols bgp]
user@R1# set local-address 172.16.1.1
user@R1# set protocols bgp export send-direct
user@R1# set group internal-peers type internal
user@R1# set group internal-peers export send-static-192.168.0
user@R1# set group internal-peers neighbor 172.16.2.2 export send-static-192.168.20
user@R1# set group internal-peers neighbor 172.16.3.3
user@R1# set group other-group type internal
user@R1# set group other-group neighbor 172.16.4.4
```

6. Configure the router ID and autonomous system (AS) number.

```
[edit routing-options]
user@R1# set router-id 172.16.1.1
user@R1# set autonomous-system 17
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@R1# commit
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
fe-1/2/0 {
    unit 0 {
        description to-R2;
        family inet {
            address 10.10.10.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 172.16.1.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    local-address 172.16.1.1;
    export send-direct;
    group internal-peers {
        type internal;
        export send-static-192.168.0;
        neighbor 172.16.2.2 {
            export send-static-192.168.20;
        }
        neighbor 172.16.3.3;
```

```
    }
    group other-group {
        type internal;
        neighbor 172.16.4.4;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-1/2/0.0;
    }
}
```

```
user@R1# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
policy-statement send-static-192.168.0 {
    term 1 {
        from {
            protocol static;
            route-filter 192.168.0.0/24 orlonger;
        }
        then accept;
    }
}
policy-statement send-static-192.168.20 {
    term 1 {
        from {
            protocol static;
            route-filter 192.168.20.0/24 orlonger;
        }
        then accept;
```

```
    }
}
```

```
user@R1# show routing-options
static {
    route 192.168.0.1/32 discard;
    route 192.168.20.1/32 discard;
}
router-id 172.16.1.1;
autonomous-system 17;
```

Verification

IN THIS SECTION

- Verifying BGP Route Learning | 36
- Verifying BGP Route Receiving | 38

Confirm that the configuration is working properly.

Verifying BGP Route Learning

Purpose

Make sure that the BGP export policies are working as expected by checking the routing tables.

Action

```
user@R1> show route protocol direct

inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.1.1/32         *[Direct/0] 1d 22:19:47
                    > via lo0.0
```

```
10.10.10.0/30      *[Direct/0] 1d 22:19:47
                    > via fe-1/2/0.0
```

```
user@R1> show route protocol static

inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.1/32     *[Static/5] 02:20:03
                      Discard
192.168.20.1/32    *[Static/5] 02:20:03
                      Discard
```

```
user@R2> show route protocol bgp
inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.20.1/32    *[BGP/170] 02:02:40, localpref 100, from 172.16.1.1
                      AS path: I, validation-state: unverified
                    > to 10.10.10.1 via fe-1/2/0.0
```

```
user@R3> show route protocol bgp
inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.1/32     *[BGP/170] 02:02:51, localpref 100, from 172.16.1.1
                      AS path: I, validation-state: unverified
                    > to 10.10.10.5 via fe-1/2/1.0
```

```
user@R4> show route protocol bgp
inet.0: 9 destinations, 11 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.1.1/32          [BGP/170] 1d 20:38:54, localpref 100, from 172.16.1.1
                      AS path: I, validation-state: unverified
                    > to 10.10.10.9 via fe-1/2/2.0
10.10.10.0/30       [BGP/170] 1d 20:38:54, localpref 100, from 172.16.1.1
```

```
                    AS path: I, validation-state: unverified
                  > to 10.10.10.9 via fe-1/2/2.0
```

Meaning

On Device R1, the `show route protocol direct` command displays two direct routes: 172.16.1.1/32 and 10.10.10.0/30. The `show route protocol static` command displays two static routes: 192.168.0.1/32 and 192.168.20.1/32.

On Device R2, the `show route protocol bgp` command shows that the only route that Device R2 has learned through BGP is the 192.168.20.1/32 route.

On Device R3, the `show route protocol bgp` command shows that the only route that Device R3 has learned through BGP is the 192.168.0.1/32 route.

On Device R4, the `show route protocol bgp` command shows that the only routes that Device R4 has learned through BGP are the 172.16.1.1/32 and 10.10.10.0/30 routes.

Verifying BGP Route Receiving

Purpose

Make sure that the BGP export policies are working as expected by checking the BGP routes received from Device R1.

Action

```
user@R2> show route receive-protocol bgp 172.16.1.1

inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 192.168.20.1/32         172.16.1.1                      100        I
```

```
user@R3> show route receive-protocol bgp 172.16.1.1

inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
```

```
  Prefix                  Nexthop              MED     Lclpref    AS path
* 192.168.0.1/32          172.16.1.1                      100        I
```

```
user@R4> show route receive-protocol bgp 172.16.1.1

inet.0: 9 destinations, 11 routes (9 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  172.16.1.1/32              172.16.1.1                      100        I
  10.10.10.0/30           172.16.1.1                      100        I
```

Meaning

On Device R2, the `route receive-protocol bgp 172.16.1.1` command shows that Device R2 received only one BGP route, 192.168.20.1/32, from Device R1.

On Device R3, the `route receive-protocol bgp 172.16.1.1` command shows that Device R3 received only one BGP route, 192.168.0.1/32, from Device R1.

On Device R4, the `route receive-protocol bgp 172.16.1.1` command shows that Device R4 received two BGP routes, 172.16.1.1/32 and 10.10.10.0/30, from Device R1.

In summary, when multiple policies are applied at different CLI hierarchies in BGP, only the most specific application is evaluated, to the exclusion of other, less specific policy applications. Although this point might seem to make sense, it is easily forgotten during router configuration, when you mistakenly believe that a neighbor-level policy is combined with a global or group-level policy, only to find that your policy behavior is not as anticipated.

RELATED DOCUMENTATION

Example: Configuring Policy Chains and Route Filters | **258**

Example: Configuring a Policy Subroutine | **289**

Example: Configuring Routing Policy Prefix Lists | **396**

export

import

Default Routing Policies

IN THIS SECTION

- OSPF and IS-IS Import Policies | 42
- Automatic Export | 43

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.

You must be familiar with the default routing policies to know when you need to modify them to suit your needs. Table 4 on page 40 summarizes the default routing policies for each routing protocol that imports and exports routes. The actions in the default routing policies are taken if you have not explicitly configured a routing policy. This table also shows direct and explicitly configured routes, which for the purposes of this table are considered a pseudoprotocol. Explicitly configured routes include aggregate, generated, and static routes.

Table 4: Default Import and Export Policies for Protocols

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
BGP	Accept all received BGP IPv4 routes learned from configured neighbors and import into the inet.0 routing table. Accept all received BGP IPv6 routes learned from configured neighbors and import into the inet6.0 routing table.	Readvertise all active BGP routes to all BGP speakers, while following protocol-specific rules that prohibit one IBGP speaker from readvertising routes learned from another IBGP speaker, unless it is functioning as a route reflector.
DVMRP	Accept all DVMRP routes and import into the inet.1 routing table.	Accept and export active DVMRP routes.

Table 4: Default Import and Export Policies for Protocols *(Continued)*

Importing or Exporting Protocol	Default Import Policy	Default Export Policy
IGMP	Import: accept all groups (regardless of being attached to an interface). In IGMP, there is no "export" from the routing table into IGMP.	
IS-IS	Accept all IS-IS routes and import into the inet.0 and inet6.0 routing tables. More information is available here: import (Protocols IS-IS)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
LDP	Accept all LDP routes and import into the inet.3 routing table.	Reject everything.
MPLS	Accept all MPLS routes and import into the inet.3 routing table.	Accept and export active MPLS routes.
OSPF	Accept all OSPF routes and import into the inet.0 routing table. (You cannot override or change this default policy.)	Reject everything. (The protocol uses flooding to announce local routes and any learned routes.)
PIM dense mode	Accept all PIM dense mode routes and import into the inet.1 routing table.	Accept active PIM dense mode routes.
PIM sparse mode	Accept all PIM sparse mode routes and import into the inet.1 routing table.	Accept and export active PIM sparse mode routes.

Table 4: Default Import and Export Policies for Protocols *(Continued)*

Importing or Exporting Protocol	Default Import Policy	Default Export Policy

Pseudoprotocol:
• Direct routes
• Explicitly configured routes:
  • Aggregate routes
  • Generated routes
  • Static routes
	Accept all direct and explicitly configured routes and import into the inet.0 routing table.	The pseudoprotocol cannot export any routes from the routing table because it is not a routing protocol.
		Routing protocols can export these or any routes from the routing table.

RIP	Accept all RIP routes learned from configured neighbors and import into the inet.0 routing table.	Reject everything. To export RIP routes, you must configure an export policy for RIP.
RIPng	Accept all RIPng routes learned from configured neighbors and import into the inet6.0 routing table.	Reject everything. To export RIPng routes, you must configure an export policy for RIPng.
Test policy	Accept all routes. For additional information about test policy, see "Example: Testing a Routing Policy with Complex Regular Expressions" on page 760.	

OSPF and IS-IS Import Policies

For OSPF, import policies apply to external routes only. An external route is a route that is outside the OSPF autonomous system (AS). For internal routes (routes learned from OSPF), you cannot change the default import policy for 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.

Automatic Export

For Layer 3 VPNs, the automatic export feature can be configured to overcome the limitation of local prefix leaking and automatically export routes between local VPN routing and forwarding (VRF) routing instances.

In Layer 3 VPNs, multiple CE routers can belong to a single VRF routing instance on a PE router. A PE router can have multiple VRF routing instances. In some cases, shared services might require routes to be written to multiple VRF routing tables, both at the local and remote PE router. This requires the PE router to share route information among each configured VRF routing instance. This exchange of route information is accomplished with custom `vrf-export` and `vrf-import` policies that utilize BGP extended community attributes to create hub-and-spoke topologies. This exchange of routing information, such as route prefixes, is known as prefix leaking.

The automatic export feature leaks prefixes between VRF routing instances that are locally configured on a given PE router. The automatic export feature is enabled by using the `auto-export` statement.

Automatic export is always applied on the local PE router, because it takes care of only local prefix leaking by evaluating the export policy of each VRF and determining which route targets can be leaked locally. The standard VRF import and export policies still affect only the remote PE prefix leaking.

If the `vrf-export` policy examined by the automatic export does not have an explicit `then accept` action, the automatic export essentially ignores the policy and, therefore, does not leak the route targets specified within it.

RELATED DOCUMENTATION

Protocol Support for Import and Export Policies | **26**

Technology Overview: Understanding the Auto Export Feature

Example: Configuring a Conditional Default Route Policy

IN THIS SECTION

- Requirements | 44
- Overview | 44
- Configuration | 45
- Verification | 50

This example shows how to configure a conditional default route on one routing device and redistribute the default route into OSPF.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 45

In this example, OSPF area 0 contains three routing devices. Device R3 has a BGP session with an external peer, for example, an Internet Service Provider (ISP).

To propagate a static route into BGP, this example includes the `discard` statement when defining the route. The ISP injects a default static route into BGP, which provides the customer network with a default static route to reach external networks. The static route has a discard next hop. This means that if a packet does not match a more specific route, the packet is rejected and a reject route for this destination is installed in the routing table, but Internet Control Message Protocol (ICMP) unreachable messages are not sent. The discard next hop allows you to originate a summary route, which can be advertised through dynamic routing protocols.

Device R3 exports the default route into OSPF. The route policy on Device R3 is conditional such that if the connection to the ISP goes down, the default route is no longer exported into OSPF because it is no

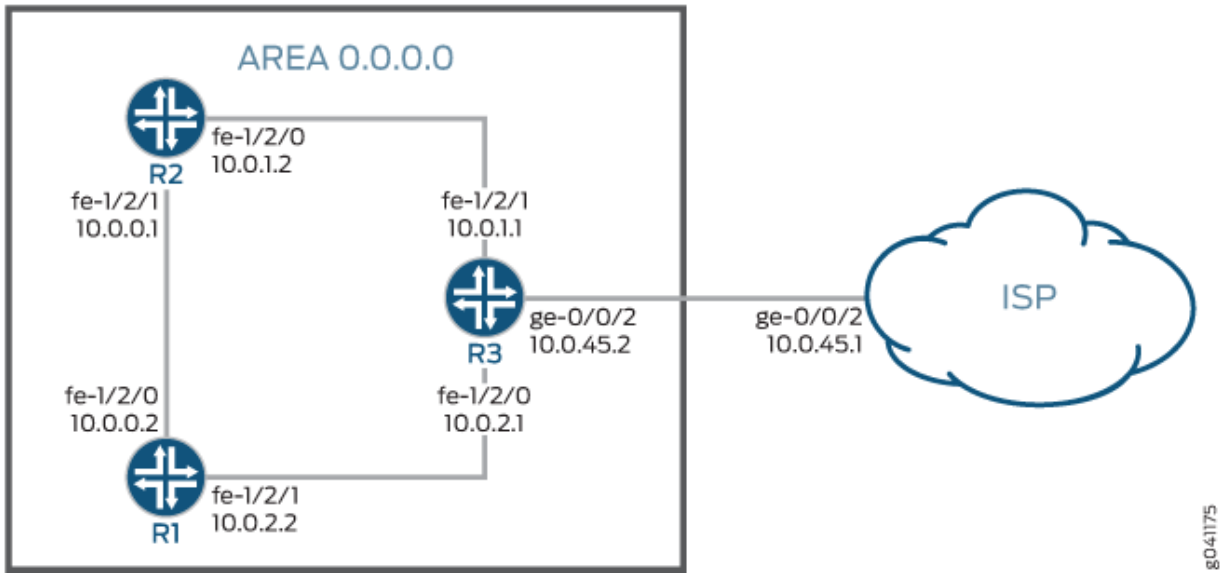
longer active in the routing table. This policy prevents packets from being silently dropped without notification (also known as null-route filtering).

This example shows the configuration for all of the devices and the step-by-step configuration on Device R3.

Topology

Figure 7 on page 45 shows the sample network.

Figure 7: OSPF with a Conditional Default Route to an ISP



Configuration

IN THIS SECTION

- CLI Quick Configuration | 46
- Procedure | 47
- Results | 49

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 description R1->R3
set interfaces fe-1/2/0 unit 0 family inet address 10.0.1.2/30
set interfaces fe-1/2/1 unit 2 description R1->R2
set interfaces fe-1/2/1 unit 2 family inet address 10.0.0.1/30
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.2
```

Device R2

```
set interfaces fe-1/2/0 unit 1 description R2->R1
set interfaces fe-1/2/0 unit 1 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 4 description R2->R3
set interfaces fe-1/2/1 unit 4 family inet address 10.0.2.2/30
set protocols ospf area 0.0.0.0 interface fe-1/2/0.1
set protocols ospf area 0.0.0.0 interface fe-1/2/1.4
```

Device R3

```
set interfaces fe-1/2/0 unit 3 description R3->R2
set interfaces fe-1/2/0 unit 3 family inet address 10.0.2.1/30
set interfaces fe-1/2/1 unit 5 description R3->R1
set interfaces fe-1/2/1 unit 5 family inet address 10.0.1.1/30
set interfaces ge-0/0/2 unit 0 description R3->ISP
set interfaces ge-0/0/2 unit 0 family inet address 10.0.45.2/30
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64500
set protocols bgp group ext neighbor 10.0.45.1
set protocols ospf export gendefault
set protocols ospf area 0.0.0.0 interface fe-1/2/1.4
set protocols ospf area 0.0.0.0 interface fe-1/2/0.3
set policy-options policy-statement gendefault term upstreamroutes from protocol bgp
set policy-options policy-statement gendefault term upstreamroutes from as-path upstream
set policy-options policy-statement gendefault term upstreamroutes from route-filter 0.0.0.0/0
```

```
upto /16
set policy-options policy-statement gendefault term upstreamroutes then next-hop 10.0.45.1
set policy-options policy-statement gendefault term upstreamroutes then accept
set policy-options policy-statement gendefault term end then reject
set policy-options as-path upstream "^64500 "
set routing-options autonomous-system 64501
```

Device ISP

```
set interfaces ge-0/0/2 unit 0 family inet address 10.0.45.1/30
set protocols bgp group ext type external
set protocols bgp group ext export advertise-default
set protocols bgp group ext peer-as 64501
set protocols bgp group ext neighbor 10.0.45.2
set policy-options policy-statement advertise-default term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement advertise-default term 1 then accept
set routing-options static route 0.0.0.0/0 discard
set routing-options autonomous-system 64500
```

Procedure

Step-by-Step Procedure

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

To configure Device R3:

1. Configure the interfaces.

```
[edit interfaces]
user@R3# set fe-1/2/0 unit 3 description R3->R2
user@R3# set fe-1/2/0 unit 3 family inet address 10.0.2.1/30
user@R3# set fe-1/2/1 unit 5 description R3->R1
user@R3# set fe-1/2/1 unit 5 family inet address 10.0.1.1/30
user@R3# set ge-0/0/2 unit 0 description R3->ISP
user@R3# set ge-0/0/2 unit 0 family inet address 10.0.45.2/30
```

2. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R3# set autonomous-system 64501
```

3. Configure the BGP session with the ISP device.

```
[edit protocols bgp group ext]
user@R3# set type external
user@R3# set peer-as 64500
user@R3# set neighbor 10.0.45.1
```

4. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R3# set interface fe-1/2/1.4
user@R3# set interface fe-1/2/0.3
```

5. Configure the routing policy.

```
[edit policy-options policy-statement gendefault]
user@R3# set term upstreamroutes from protocol bgp
user@R3# set term upstreamroutes from as-path upstream
user@R3# set term upstreamroutes from route-filter 0.0.0.0/0 upto /16
user@R3# set term upstreamroutes then next-hop 10.0.45.1
user@R3# set term upstreamroutes then accept
user@R3# set term end then reject
[edit policy-options]
user@R3# set as-path upstream "^64500 "
```

6. Apply the export policy to OSPF.

```
[edit protocols ospf]
user@R3# set export gendefault
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@R3# commit
```

Results

Confirm your configuration by issuing the `show` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show
interfaces {
    fe-1/2/0 {
        unit 3 {
            description R3->R2;
            family inet {
                address 10.0.2.1/30;
            }
        }
    }
    fe-1/2/1 {
        unit 5 {
            description R3->R1;
            family inet {
                address 10.0.1.1/30;
            }
        }
    }
    ge-1/2/0 {
        unit 0 {
            description R3->ISP;
            family inet {
                address 10.0.45.2/30;
            }
        }
    }
}
protocols {
    bgp {
        group ext {
            type external;
```

```
            peer-as 64500;
            neighbor 10.0.45.1;
        }
    }
    ospf {
        export gendefault;
        area 0.0.0.0 {
            interface fe-1/2/1.4;
            interface fe-1/2/0.3;
        }
    }
}
policy-options {
    policy-statement gendefault {
        term upstreamroutes {
            from {
                protocol bgp;
                as-path upstream;
                route-filter 0.0.0.0/0 upto /16;
            }
            then {
                next-hop 10.0.45.1;
                accept;
            }
        }
        term end {
            then reject;
        }
    }
    as-path upstream "^64500 ";
}
routing-options {
    autonomous-system 64501;
}
```

Verification

IN THIS SECTION

Verifying That the Route to the ISP Is Working | 51

Verifying That the Static Route Is Redistributed | 51
Testing the Policy Condition | 53

Confirm that the configuration is working properly.

Verifying That the Route to the ISP Is Working

Purpose

Make sure connectivity is established between Device R3 and the ISP's router.

Action

```
user@R3> ping 10.0.45.1
PING 10.0.45.1 (10.0.45.1): 56 data bytes
64 bytes from 10.0.45.1: icmp_seq=0 ttl=64 time=1.185 ms
64 bytes from 10.0.45.1: icmp_seq=1 ttl=64 time=1.199 ms
64 bytes from 10.0.45.1: icmp_seq=2 ttl=64 time=1.186 ms
```

Meaning

The `ping` command confirms reachability.

Verifying That the Static Route Is Redistributed

Purpose

Make sure that the BGP policy is redistributing the static route into Device R3's routing table. Also make sure that the OSPF policy is redistributing the static route into the routing tables of Device R1 and Device R2.

Action

```
user@R3> show route protocol bgp

inet.0: 9 destinations, 10 routes (9 active, 0 holddown, 1 hidden)
```

```
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:00:25, localpref 100
                      AS path: 64500 I
                    > to 10.0.45.1 via ge-0/0/2.6
```

```
user@R1> show route protocol ospf

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/150] 00:03:58, metric 0, tag 0
                    > to 10.0.1.1 via fe-1/2/0.0
10.0.2.0/30        *[OSPF/10] 03:37:45, metric 2
                      to 10.0.1.1 via fe-1/2/0.0
                    > to 10.0.0.2 via fe-1/2/1.2
172.16.233.5/32       *[OSPF/10] 03:38:41, metric 1
                      MultiRecv
```

```
user@R2> show route protocol ospf
inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/150] 00:04:04, metric 0, tag 0
                    > to 10.0.2.1 via fe-1/2/1.4
10.0.1.0/30        *[OSPF/10] 03:37:46, metric 2
                      to 10.0.0.1 via fe-1/2/0.1
                    > to 10.0.2.1 via fe-1/2/1.4
172.16.233.5/32       *[OSPF/10] 03:38:47, metric 1
                      MultiRecv
```

Meaning

The routing tables contain the default 0.0.0.0/0 route. If Device R1 and Device R2 receive packets destined for networks not specified in their routing tables, those packets will be sent to Device R3 for further processing. If Device R3 receives packets destined for networks not specified in its routing table, those packets will be sent to the ISP for further processing.

Testing the Policy Condition

Purpose

Deactivate the interface to make sure that the route is removed from the routing tables if the external network becomes unreachable.

Action

```
user@R3> deactivate interfaces ge-0/0/2 unit 0 family inet address 10.0.45.2/30
user@R3> commit
```

```
user@R1> show route protocol ospf

inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.2.0/30        *[OSPF/10] 03:41:48, metric 2
                      to 10.0.1.1 via fe-1/2/0.0
                    > to 10.0.0.2 via fe-1/2/1.2
172.16.233.5/32       *[OSPF/10] 03:42:44, metric 1
                      MultiRecv

user@R2> show route protocol ospf
inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.1.0/30        *[OSPF/10] 03:42:10, metric 2
                      to 10.0.0.1 via fe-1/2/0.1
                    > to 10.0.2.1 via fe-1/2/1.4
172.16.233.5/32       *[OSPF/10] 03:43:11, metric 1
                      MultiRecv
```

Meaning

The routing tables on Device R1 and Device R2 do not contain the default 0.0.0.0/0 route. This verifies that the default route is no longer present in the OSPF domain. To reactivate the ge-0/0/2.6 interface, issue the `activate interfaces ge-0/0/2 unit 0 family inet address 10.0.45.2/30` configuration mode command.

CHAPTER 2

Evaluating Routing Policies Using Match Conditions, Actions, Terms, and Expressions

IN THIS CHAPTER

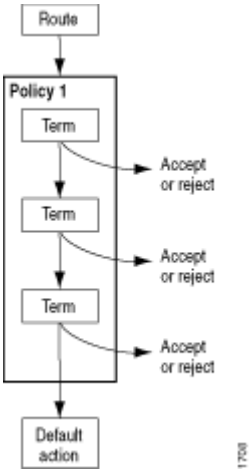
- How a Routing Policy Is Evaluated | 54
- Categories of Routing Policy Match Conditions | 56
- Routing Policy Match Conditions | 58
- Route Filter Match Conditions | 72
- Actions in Routing Policy Terms | 75
- Summary of Routing Policy Actions | 92
- Example: Configuring a Routing Policy to Advertise the Best External Route to Internal Peers | 95
- Example: Configuring BGP to Advertise Inactive Routes | 107
- Example: Using Routing Policy to Set a Preference Value for BGP Routes | 118
- Example: Enabling BGP Route Advertisements | 126
- Example: Rejecting Known Invalid Routes | 137
- Example: Using Routing Policy in an ISP Network | 141
- Understanding Policy Expressions | 204
- Understanding Backup Selection Policy for OSPF Protocol | 210
- Configuring Backup Selection Policy for the OSPF Protocol | 211
- Configuring Backup Selection Policy for IS-IS Protocol | 219
- Example: Configuring Backup Selection Policy for the OSPF or OSPF3 Protocol | 221

How a Routing Policy Is Evaluated

Figure 8 on page 55 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" on page 40.

Figure 8: Routing Policy Evaluation



Each term can consist of two statements, `from` and `to`, that define match conditions:

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, all conditions must match the route for a match to occur.

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, all conditions must match the route for a match to occur.

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

Categories of 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 are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions, also called named match conditions.

Extended 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.

Table 5 on page 56 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 "Routing Policy Match Conditions" on page 58.

Table 5: Match Condition Concepts

Match Condition	Category	When to Use	Notes
AS path regular expression—A combination of AS numbers and regular expression operators.	Extended	(BGP only) Match a route based on its AS path. (An AS path consists of the AS numbers of all routers a packet must go through to reach a destination.) You can specify an exact match with a particular AS path or a less precise match.	You use regular expressions to match the AS path.

Table 5: Match Condition Concepts *(Continued)*

Match Condition	Category	When to Use	Notes
Community—A group of destinations that share a property. (Community information is included as a path attribute in BGP update messages.)	Extended	Match a group of destinations that share a property. Use a routing policy to define a community that specifies a group of destinations you want to match and one or more actions that you want taken on this community.	Actions can be performed on the entire group. You can create multiple communities associated with a particular destination. You can create match conditions using regular expressions.
Prefix list—A named list of IP addresses.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route only.	You can specify a common action only for all prefixes in the list.
Route list—A list of destination prefixes.	Extended	Match a route based on prefix information. You can specify an exact match of a particular route or a less precise match.	You can specify an action for each prefix in the route list or a common action for all prefixes in the route list.
Standard—A collection of criteria that can match a route.	Standard	Match a route based on one of the following criteria: area ID, color, external route, family, instance (routing), interface name, level number, local preference, metric, neighbor address, next-hop address, origin, preference, protocol, routing table name, or tag. You can specify a match condition for policies based on protocols by naming a protocol from which the route is learned or to which the route is being advertised.	None.

Table 5: Match Condition Concepts *(Continued)*

Match Condition	Category	When to Use	Notes
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 287.

Each term can consist of two statements, `from` and `to`, that define match conditions:

- 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, all conditions must match the route for a match to occur.
- 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, all conditions must match the route for a match to occur.

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

RELATED DOCUMENTATION

Routing Policy Match Conditions | 58

Routing Policy Match Conditions

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;
}
```

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`, all routes are considered to match. All routes then take the configured actions of the policy term.

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.

Table 6 on page 59 summarizes key routing policy match conditions.

Table 6: Summary of Key Routing Policy Match Conditions

Match Condition	Description
`aggregate-contributor`	Matches routes that are contributing to a configured aggregate. This match condition can be used to suppress a contributor in an aggregate route.
`area` *area-id*	Matches a route learned from the specified OSPF area during the exporting of OSPF routes into other protocols.
`as-path` *name*	Matches the name of the path regular expression of an autonomous systems (AS). BGP routes whose AS path matches the regular expression are processed.

Table 6: Summary of Key Routing Policy Match Conditions *(Continued)*

Match Condition	Description
`color` *preference*	Matches a color value. You can specify preference values that are finer-grained than those specified in the *preference* match conditions. The `color` value can be a number from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route.
`community`	Matches the 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.)
`external [type` *metric-type*`]`	Matches external OSPF routes, including routes exported from one level to another. In this match condition, `type` is an optional keyword. The `metric-type` value can be either 1 or 2. When you do not specify `type`, this condition matches all external routes.
`interface` *interface-name*	Matches the name or IP address of one or more router interfaces. Use this condition with protocols that are interface-specific. For example, do not use this condition with internal BGP (IBGP). Depending on where the policy is applied, this match condition matches routes learned from or advertised through the specified interface.
`internal`	Matches a routing policy against the internal flag for simplified next-hop self policies.
`level` *level*	Matches the IS-IS level. Routes that are from the specified level or are being advertised to the specified level are processed.
`local-preference` *value*	Matches a BGP local preference attribute. The preference value can be from 0 through 4,294,967,295 (2^{32} – 1).
`metric` *metric* `metric2` *metric*	Matches a metric value. The `metric` value corresponds to the multiple exit discriminator (MED), and `metric2` corresponds to the IGP metric if the BGP next hop runs back through another route.

Table 6: Summary of Key Routing Policy Match Conditions *(Continued)*

Match Condition	Description
`neighbor` *address*	Matches the address of one or more neighbors (peers). For BGP export policies, the address can be for a directly connected or indirectly connected peer. For all other protocols, the address is for the neighbor from which the advertisement is received.
`next-hop` *address*	Matches the 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.
`origin` *value*	Matches the BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: • `egp`—Path information originated from another AS. • `igp`—Path information originated from within the local AS. • `incomplete`—Path information was learned by some other means.
`preference` *preference* `preference2` *preference*	Matches the preference value. You can specify a primary preference value (`preference`) and a secondary preference value (`preference2`). The preference value can be a number from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route. **NOTE**: Do not set `preference2` for BGP route-policy.
`protocol` *protocol*	Matches the 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: `aggregate`, `bgp`, `direct`, `dvmrp`, `isis`, `local`, `ospf`, `pim-dense`, `pim-sparse`, `rip`, `ripng`, or `static`.
`route-type` *value*	Matches the type of route. The value can be either `external` or `internal`.

All conditions in the `from` and `to` statements must match for the action to be taken. The match conditions defined in Table 7 on page 62 are effectively a logical AND operation. Matching in prefix lists and route lists is handled differently. They are effectively a logical OR operation. 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 7 on page 62 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 7 on page 62 also indicates whether the match condition is standard or extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (autonomous system [AS] path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions.

Table 7: Complete List of Routing Policy Match Conditions

Match Condition	Match Condition Category	Statement Description
`aggregate-contributor`	Standard	Match routes that are contributing to a configured aggregate. This match condition can be used to suppress a contributor in an aggregate route.
`area` *`area-id`*	Standard	(Open Shortest Path First [OSPF] only) Area identifier. In a `from` statement used with an `export` policy, match a route learned from the specified OSPF area when exporting OSPF routes into other protocols.
`as-path` *`name`*	Extended	(Border Gateway Protocol [BGP] only) Name of an AS path regular expression. For more information, see "Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions" on page 433.
`as-path-group` *`group-name`*	Extended	(BGP only) Name of an AS path group regular expression. For more information, see "Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions" on page 433.
`color` *`preference`* `color2` *`preference`*	Standard	Color value. You can specify preference values (`color` and `color2`) that are finer-grained than those specified in the `preference` and `preference2` match conditions. The color value can be a number in the range from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`community-count` *`value`* `(equal | orhigher | orlower)`	Standard	(BGP only) Number of community entries required for a route to match. The count value can be a number in the range of 0 through 1,024. Specify one of the following options: • `equal`—The number of communities must equal this value to be considered a match. • `orhigher` —The number of communities must be greater than or equal to this value to be considered a match. • `orlower`—The number of communities must be less than or equal to this value to be considered a match. **NOTE**: If you configure multiple `community-count` statements, the matching is effectively a logical AND operation. **NOTE**: The `community-count` attribute only works with standard communities. It does not work with extended communities. This match condition is not supported for use with the To statement.
`community` `[` *`names`* `]`	Extended	Name of one or more communities. If you list more than one name, only one name needs to match for a match to occur (the matching is effectively a logical OR operation). For more information, see *Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions*. BGP EVPN routes have a set of extended communities carried in the BGP update message path attribute, and as such, you can use extended communities for filtering BGP EVPN routes. The information available includes encapsulation type, mac-mobility information, EVPN split-horizon label information, ESI mode, and etree leaf label. Use the following syntax to specify BGP EVPN extended communities: • `community (type, in decimal format)` *`val1`*`:`*`val2`* *val1* and *val2* can be specified as [2 + 4] octets, or as [4 + 2] octets.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`external [ type` *`metric-type`* `]`	Standard	(OSPF and IS-IS only) Match IGP external routes. For IS-IS routes, the `external` condition also matches routes that are exported from one IS-IS level to another. The `type` keyword is optional and is applicable only to OSPF external routes. When you do not specify `type`, the `external` condition matches all IGP external (OSPF and IS-IS) routes. When you specify `type`, the `external` condition matches only OSPF external routes with the specified OSPF metric type. The metric type can either be 1 or 2. To match BGP external routes, use the `route-type` match condition.
`evpn-esi`	Standard	You can filter BGP EVPN routes on the basis of Ethernet Segment Identifiers (ESIs) information for routes types 1, 2, 4, 7, and 8, which are the only types to include the ESI attribute in their prefix. (ESI values are encoded as 10-byte integers and are used to identify a multihomed segment.)
`evpn-etag`	Standard	You can filter BGP EVPN routes on the basis of EVPN tag information, which is available from the prefix of the EVPN route. Requires EVPN be set in the following CLI hierarchy: • `filter policy-options policy-statement` *`name`* `term` *`name`* `family`
`evpn-mac-route`	Standard	Filtering BGP EVPN type-2 routes based on if it has any IP address. EVPN type-2 MAC routes can have IP address in the prefix along with MAC address. The IP address carried in the MAC-IP route can be either IPv4 or IPv6 address. It is possible to filter out type-2 routes based on only if it has only mac address or mac+ipv4 address or mac+ipv6 address.
`interface` *`interface-name`*	Standard	Name or IP address of one or more routing device interfaces. Do not use this qualifier with protocols that are not interface-specific, such as IBGP. Match a route learned from, or to be advertised to, one of the specified interfaces. Direct routes match routes configured on the specified interface.
`level` *`level`*	Standard	(Intermediate System-to-Intermediate System [IS-IS] only) IS-IS level. Match a route learned from, or to be advertised to, a specified level.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`local-preference` *value*	Standard	(BGP only) BGP local preference (LOCAL_PREF`local-preference (add | subtract)` *number*) attribute. The preference value can be a number in the range 0 through 4,294,967,295 (2^{32} – 1).
`mac-filter-list`	Standard	(BGP only) Named mac filter list. EVPN type-2 routes have mac address as part of the prefix, which you can use to create a list of MAC addresses.
`multicast-scoping` (*scoping-name* | *number*) < (`orhigher` | `orlower`) >	Standard	Multicast scope value of IPv4 or IPv6 multicast group address. The multicast-scoping name corresponds to an IPv4 prefix. You can match on a specific multicast-scoping prefix or on a range of prefixes. Specify `orhigher` to match on a scope and numerically higher scopes, or `orlower` to match on a scope and numerically lower scopes. For more information, see the Junos OS Multicast Protocols User Guide. You can apply this scoping policy to the routing table by including the `scope-policy` statement at the `[edit routing-options]` hierarchy level. The *number* value can be any hexadecimal number from 0 through F. The multicast-scope value is a number from 0 through 15, or one of the following keywords with the associated meanings: • `node-local` (value=1)—No corresponding prefix • `link-local` (value=2)—Corresponding prefix 224.0.0.0/24 • `site-local` (value=5)—No corresponding prefix • `global` (value=14)—Corresponding prefix 224.0.1.0 through 238.255.255.255 • `organization-local` (value=8)—Corresponding prefix 239.192.0.0/14

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`neighbor` *address*	Standard	Address of one or more neighbors (peers). For BGP, the address can be a directly connected or indirectly connected peer. For BGP *import* policies, specifying `to neighbor` produces the same result as specifying `from neighbor`. For BGP *export* policies, specifying the `neighbor` match condition has no effect and is ignored. For all other protocols, the address is the neighbor from which the advertisement is received, or for `to` statements, it matches the neighbor to which the advertisement is sent. **NOTE**: The `neighbor` *address* match condition is not valid for the Routing Information Protocol (RIP).
`next-hop` [*addresses*]	Standard	One or more next-hop addresses specified in the routing information for a particular route. A next-hop address cannot include a netmask. For BGP routes, matches are performed against each protocol next hop.
`next-hop-type merged`	Standard	LDP generates a next hop based on RSVP and IP next hops available to use, combined with forwarding-class mapping. This match condition is not supported for use with the To statement.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`nlri-route-type`	Standard	Route type from NLRI 1 through NLRI 10. Multiple route types can be specified in a single policy. For EVPN, NLRI route types range from 1 to 8 (the first octet of the route prefix in the BGP update message is the EVPN route type). In addition to filtering on EVPN NLRI route types, you can also filter on IP address or MAC address (mac-ip) that is embedded in the EVPN route prefix for route types 2 and 5. To do so, use a `prefix-list` or `route-filter` for the address. When a type-5 route is created from a type 2 mac-ip advertisement that was learned remotely, then the community that was learned from the type-2 route advertisement is included in the new type-5 route. You can prevent this by enabling the `donot-advertise-community` statement at the `protocols evpn ip-prefix-routes` hierarchy.
`origin` *value*	Standard	(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: • `egp`—Path information originated in another AS. • `igp`—Path information originated within the local AS. • `incomplete`—Path information was learned by some other means.
`policy [` *policy-name* `]`	Extended	Name of a policy to evaluate as a subroutine. For information about this extended match condition, see "Understanding Policy Subroutines in Routing Policy Match Conditions" on page 283.
`preference` *preference* `preference2` *preference*	Standard	Preference value. You can specify a primary preference value (`preference`) and a secondary preference value (`preference2`). The preference value can be a number from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route. To specify even finer-grained preference values, see the `color` and `color2` match conditions in this table.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`prefix-list` *`prefix-list-name ip-addresses`*	Extended	Named list of IP addresses. You can specify an exact match with incoming routes. For information about this extended match condition, see "Understanding Prefix Lists for Use in Routing Policy Match Conditions" on page 392. This match condition is not supported for use with the To statement. This match condition is not supported for use with the To statement.
`prefix-list-filter` *`prefix-list-name match-type`*	Extended	Named prefix list. You can specify prefix length qualifiers for the list of prefixes in the prefix list. When used with EVPN NRLI route types 2 and 5, the following are supported: • `from prefix-list-filter [ exact | longer | orlonger ]` For information about this extended match condition, see "Understanding Prefix Lists for Use in Routing Policy Match Conditions" on page 392. This match condition is not supported for use with the To statement.
`protocol` *`protocol`*	Standard	Name of the protocol from which the route was learned or to which the route is being advertised. It can be one of the following: `access`, `access-internal`, `aggregate`, `anchor`, `arp`, `bgp`, `bgp-ls-epe`,`bgp-static`, `direct`, `dvmrp`, `esis`, `evpn`, `frr`, `isis`, `l-isis`, `isis`, `l2-learned-host-routing`, `l2circuit`, `l2vpn`, `ldp`, `local`, `mpls`, `msdp`, `ospf` (matches both OSPFv2 and OSPFv3 routes), `ospf2` (matches OSPFv2 routes only), `ospf3` (matches OSPFv3 routes only), `pim`, `rift`, `rip`, `ripng`, `route-target`, `rsvp`, `spring-te`, `static`, or `vpls`.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`rib routing-table`	Standard	Name of a routing table. The value of *`routing-table`* can be one of the following: • inet.0—Unicast IPv4 routes • *instance-name* inet.0—Unicast IPv4 routes for a particular routing instance • inet.1—Multicast IPv4 routes • inet.2—Unicast IPv4 routes for multicast reverse-path forwarding (RPF) lookup • inet.3—MPLS routes • mpls.0—MPLS routes for label-switched path (LSP) next hops • inet6.0—Unicast IPv6 routes
`route-distinguisher`	Standard	Name of the route-distinguisher (RD). RD supports filtering BGP EVPN routes. The RD information is carried in the prefix of the EVPN route.
`route-filter route-filter-list`	Standard	Named route filter or route filter list. You can specify prefix length qualifiers for the list of routes in the route filter list. When used with EVPN NRLI route types 2 and 5, the following are supported: • `from route-filter [ address-mask | exact | longer | orlonger | prefix-length-range | through | upto ]` This match condition is not supported for use with the To statement.
`rtf-prefix-list name route-targets`	Extended	(BGP only) Named list of route target prefixes for BGP route target filtering and proxy BGP route target filtering. For information about this extended match condition, see *Example: Configuring an Export Policy for BGP Route Target Filtering for VPNs*. This match condition is not supported for use with the To statement.

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`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 "Understanding Route Filters for Use in Routing Policy Match Conditions" on page 303. This match condition is not supported for use with the To statement.
`state (active | inactive)`	Standard	(BGP export only) Match on the following types of advertised routes: • `active`—An active BGP route • `inactive`—A route advertised to internal BGP peers as the best external path even if the best path is an internal route • `inactive`—A route advertised by BGP as the best route even if the routing table did not select it to be an active route

Table 7: Complete List of Routing Policy Match Conditions *(Continued)*

Match Condition	Match Condition Category	Statement Description
`tag` *`string`* `tag2` *`string`*	Standard	Tag value. You can specify two tag strings: `tag` (for the first string) and `tag2`. These values are local to the router and can be set on configured routes or by using an import routing policy. You can specify multiple tags under one match condition by including the tags within a bracketed list. For example: `from tag [ tag1 tag2 tag3 ];` For OSPF routes, the`tag` action sets the 32-bit tag field in OSPF external link-state advertisement (LSA) packets. For IS-IS routes, the `tag` action sets the 32-bit flag in the IS-IS IP prefix type length values. (TLV). OSPF stores the INTERNAL route's OSPF area ID in the`tag2` attribute. However, for EXTERNAL routes, OSPF does not store anything in the `tag2`attribute. You can configure a policy term to set the `tag2` value for a route. If the route, already has a `tag2` value (for example, an OSPF route that stores area id in tag2), then the original `tag2` value is overwritten by the new value. When the policy contains the "from area" match condition, for internal OSPF routes, where `tag2` is set, based on the OSPF area- ID, the evaluation is conducted to compare the `tag2` attribute with the area ID. For external OSPF routes that do not have the `tag2` attribute set, the match condition fails.
`validation-database`	Standard	When BGP origin validation is configured, triggers a lookup in the route validation database to determine if the route prefix is valid, invalid, or unknown. The route validation database contains route origin authorization (ROA) records that map route prefixes to expected originating autonomous systems (ASs). This prevents the accidental advertisement of invalid routes. See Configuring Origin Validation for BGP.

RELATED DOCUMENTATION

Understanding Prefix Lists for Use in Routing Policy Match Conditions | 392

Understanding Route Filters for Use in Routing Policy Match Conditions | 303

Route Filter Match Conditions

When specifying a destination prefix, you can specify an exact match with a specific route, or a less precise match by using match types. You can configure either a common reject action that applies to the entire list, or an action associated with each prefix.

You can specify known invalid (“bad”) routes to ignore by specifying matches on destination prefixes. Additionally, you can specify that “good” routes be processed in a particular way. For instance, you can group traffic from specific source or destination addresses into forwarding classes to be processed using the *class of service* (CoS) feature.

Table 8 on page 73 lists route list match types.

Table 8: Route List Match Types

Match Type	Match Conditions
address-mask *netmask-value*	All of the following are true: • The bit-wise logical AND of the *netmask-value* pattern and the incoming IPv4 or IPv6 route address and the bit-wise logical AND of the *netmask-value* pattern and the *destination-prefix* address are the same. The bits set in the *netmask-value* pattern do not need to be contiguous. • The *prefix-length* component of the incoming IPv4 or IPv6 route address and the *prefix-length* component of the *destination-prefix* address are the same. **NOTE**: The address-mask routing policy match type is valid only for matching an incoming IPv4 (family inet) or IPv6 (family inet6) route address to a list of destination match prefixes specified in a route-filter statement. The address-mask routing policy match type enables you to match an incoming IPv4 or IPv6 route address on a configured netmask address in addition to the length of a configured destination match prefix. The length of the route address must match exactly with the length of the configured destination match prefix, as the address-mask match type does not support prefix length variations for a range of prefix lengths. When the longest-match lookup is performed on a route filter, the lookup evaluates an address-mask match type differently from other routing policy match types. The lookup does not consider the length of the destination match prefix. Instead, the lookup considers the number of contiguous high-order bits set in the netmask value.
exact	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is equal to the route's prefix length.
longer	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is greater than the route's prefix length.

Table 8: Route List Match Types *(Continued)*

Match Type	Match Conditions
`orlonger`	The route shares the same most-significant bits (described by *prefix-length*), and *prefix-length* is equal to or greater than the route's prefix length.
`prefix-length-range` *prefix-length2-prefix-length3*	The route shares the same most-significant bits (described by *prefix-length*), and the route's prefix length falls between *prefix-length2* and *prefix-length3*, inclusive.
`through` *destination-prefix*	All the following are true: • The route shares the same most-significant bits (described by *prefix-length*) of the first destination prefix. • The route shares the same most-significant bits (described by *prefix-length*) 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. You do not use the `through` match type in most routing policy configurations.
`upto` *prefix-length2*	The route shares the same most-significant bits (described by *prefix-length*) and the route's prefix length falls between *prefix-length* and *prefix-length2*.

RELATED DOCUMENTATION

- Categories of Routing Policy Match Conditions | **56**
- Summary of Routing Policy Actions | **92**
- Example: Rejecting Known Invalid Routes | **137**
- Example: Grouping Source and Destination Prefixes into a Forwarding Class | **671**

Actions in Routing Policy Terms

IN THIS SECTION

- Configuring Flow Control Actions | 76
- Configuring Actions That Manipulate Route Characteristics | 77
- Configuring the Default Action in Routing Policies | 88
- Configuring a Final Action in Routing Policies | 90
- Logging Matches to a Routing Policy Term | 91
- Configuring Separate Actions for Routes in Route Lists | 91

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 "Routing Policy Match Conditions" on page 58.

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 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" on page 40.

The following sections discuss these actions:

Configuring Flow Control Actions

Table 9 on page 76 lists the flow control actions. You can specify one of these actions along with the trace action or one or more of the actions that manipulate route characteristics (see "Configuring Actions That Manipulate Route Characteristics" on page 77).

Table 9: 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.

Table 9: Flow Control Actions *(Continued)*

Flow Control Action	Description
`default-action reject`	Reject and override any action intrinsic to the protocol. This is a nonterminating policy action.
`next term`	Skip to and evaluate the next term in the same routing policy. Any accept or reject action specified in the `then` statement is skipped. Any actions in the `then` statement that manipulate route characteristics are applied to the route. `next term` is the default control action if a match occurs and you do not specify a flow control action. **NOTE**: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.
`next policy`	Skip to and evaluate the next routing policy. Any accept or reject action specified in the `then` statement is skipped. Any actions in the `then` statement that manipulate route characteristics are applied to the route. `next policy` is the default control action if a match occurs, you do not specify a flow control action, and there are no further terms in the current routing policy.
sr-te-template	Segment routing-traffic engineered (SR-TE) template to apply for PCE-initiated LSPs.

Configuring Actions That Manipulate Route Characteristics

You can specify one or more of the actions listed in Table 10 on page 77 to manipulate route characteristics.

Table 10: Actions That Manipulate Route Characteristics

Action	Description
`add-path send-count` *`path-count`*	(BGP only) Enable sending up to 20 BGP paths to a destination for a subset of `add-path` advertised prefixes.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`as-path-prepend` *as-path*	(BGP only) Affix one or more AS numbers at the beginning of the AS path. If specifying more than one AS number, enclose the numbers in quotation marks (“ ”). The AS numbers are added after the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed within a nonconfederation sequence. For more information, see "Understanding Prepending AS Numbers to BGP AS Paths" on page 457. In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS.
`as-path-expand last-as count` *n*	(BGP only) Extract the last AS number in the existing AS path and affix that AS number to the beginning of the AS path *n* times, where *n* is a number from 1 through 32. The AS number is added before the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed within a non-confederation sequence. This option is typically used in non-IBGP export policies. **NOTE**: Starting in Junos OS Release 17.3, it is possible to commit a null configuration for the count value, and if so, Junos will convert the null to a 1 count rather than a 0 count, or disallowing the commit. The effect of having your `as-path-expand` count equal one is that such an *as-path* is longer, and therefore less preferable. We recommend that you either explicitly set the `as-path-expand` count, or delete the unused setting to avoid any unexpected behavior.
bgp-output-queue-priority	(BGP only) Set the output priority queue used for this route. There are 17 prioritized output queues: an expedited queue that is the highest priority, and 16 numbered queues where 1 is the lowest priority and 16 is the highest.
`class` *class-name*	(Class of service [CoS] only) Apply the specified class-of-service parameters to routes installed into the routing table. For more information, see the Junos OS Class of Service User Guide for Routing Devices.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`color` *preference* `color2` *preference*	Set the preference value to the specified value. The `color` and `color2` preference values are even more fine-grained than those specified in the `preference` and `preference2` actions. The color value can be a number in the range from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route. If you set the preference with the `color` action, the value is internal to Junos OS and is not transitive.
`color (add | subtract)` *number* `color2 (add | subtract)` *number*	Change the color preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 (2^{32} – 1), the value is set to 2^{32} – 1. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
`community (+ | add) [` *names* `]`	(BGP only) Add the specified communities to the set of communities in the route. For more information, see *Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions*.
`community (- | delete) [` *names* `]`	(BGP only) Delete the specified communities from the set of communities in the route. For more information, see *Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions*.
`community (= | set) [` *names* `]`	(BGP only) Replace any communities that were in the route in with the specified communities. For more information, see *Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions*.
`cos-next-hop-map` *map-name*	Set CoS-based next-hop map in forwarding table.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`damping` *name*	(BGP only) Apply the specified route-damping parameters to the route. These parameters override the default damping parameters. This action is useful only in an import policy, because the damping parameters affect the state of routes in the routing table. To apply damping parameters, you must enable BGP flap damping as described in the Junos OS Routing Protocols Library for Routing Devices, and you must create a named list of parameters as described in "Using Routing Policies to Damp BGP Route Flapping" on page 598.
`destination-class` *destination-class-name*	Maintain packet counts for a route passing through your network, based on the destination address in the packet. You can do the following: • Configure group destination prefixes by configuring a routing policy. • Apply that routing policy to the forwarding table with the corresponding destination class. • Enable packet counting on one or more interfaces by including the `destination-class-usage` statement at the `[edit interfaces` *interface-name* `unit` *logical-unit-number* `family inet accounting]` hierarchy level (see the Junos OS Class of Service User Guide for Routing Devices). • View the output by using one of the following commands: `show interfaces destination-class (all |` *destination-class-name logical-interface-name*), `show interfaces` *interface-name* `extensive`, or `show interfaces` *interface-name* `statistics` (see the CLI Explorer). • To configure a packet count based on the source address, use the `source-class` statement described in this table.
`external type` *metric*	Set the external metric type for routes exported by OSPF. You must specify the keyword `type`.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`forwarding-class` *`forwarding-class-name`*	Create the forwarding class that includes packets based on both the destination address and the source address in the packet. You can do the following: • Configure group prefixes by configuring a routing policy. • Apply that routing policy to the forwarding table with the corresponding forwarding class. • Enable packet counting on one or more interfaces by using the procedure described in either the `destination-class` or `source-class` actions defined in this table.
`install-nexthop <strict> lsp` *`lsp-name`*	Choose which next hops, among a set of equal LSP next hops, are installed in the forwarding table. Use the export policy for the forwarding table to specify the LSP next hop to be used for the desired routes. Specify the `strict` option to enable strict mode, which checks to see if any of the LSP next hops specified in the policy are up. If none of the specified LSP next hops are up, the policy installs the discard next hop.
`install-to-fib`	For PTX Series routers only, override the default BGP routing policy. For more information, see Example: Overriding the Default BGP Routing Policy on PTX Series Packet Transport Routers.
`load-balance consistent-hash`	(BGP only) For MX Series routers with modular port concentrators (MPCs) and for QFX10000 switches only, specify consistent load balancing for one or more IP addresses. This feature preserves the affinity of a flow to a path in an equal-cost multipath (ECMP) group when one or more next-hop paths fail. Only flows for paths that are inactive are redirected. Flows mapped to servers that remain active are maintained.
`load-balance destination-ip-only`	Calculate load balancing hash based solely on destination IP address. This allows a service provider to direct traffic toward a specific content server in per-subscriber aware environments.
`load-balance per-packet`	(For export to the forwarding table only) Install all next-hop addresses in the forwarding table and have the forwarding table perform per-packet load balancing. This policy action allows you to optimize VPLS traffic flows across multiple paths. For more information, see *Configuring Per-Packet Load Balancing*.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`load-balance per-prefix`	For PTX Series routers only, override the default per-packet load balancing routing policy for BGP. For more information, see Example: Overriding the Default BGP Routing Policy on PTX Series Packet Transport Routers.
`load-balance source-ip-only`	Calculate load balancing hash based solely on source IP address. This allows a service provider to direct traffic toward a specific content server in per-subscriber aware environments.
`local-preference` *value*	(BGP only) Set the BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range from 0 through 4,294,967,295 (2^{32} – 1).
`local-preference (add | subtract)` *number*	Change the local preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 (2^{32} – 1), the value is set to 2^{32} – 1. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value. For BGP, if the attribute value is not known, it is initialized to 100 before the routing policy is applied.
`map-to-interface (`*interface-name* `| self)`	Sets the `map-to-interface` value which is similar to existing metric or tag actions. The `map-to-interface` action requires you to specify one of the following: • A logical interface (for example, ge-0/0/0.0). The logical interface can be any interface that multicast currently supports, including VLAN and aggregated Ethernet interfaces. **NOTE**: If you specify a physical interface as the `map-to-interface` (for example, ge-0/0/0), a value of .0 is appended to physical interface to create a logical interface. • The keyword `self`. The `self` keyword specifies that multicast data packets are sent on the same interface as the control packets and no mapping occurs. If no term matches, then no multicast data packets are sent.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`metric` *metric* `metric2` *metric* `metric3` *metric* `metric4` *metric*	Set the metric. You can specify up to four metric values, starting with `metric` (for the first metric value) and continuing with `metric2`, `metric3`, and `metric4`. (BGP only) `metric` corresponds to the MED, and `metric2` corresponds to the IGP metric if the BGP next hop loops through another router.
`metric (add | subtract)` *number* `metric2 (add | subtract)` *number* `metric3 (add | subtract)` *number* `metric4 (add | subtract)` *number*	Change the metric value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
`metric expression (metric multiplier` *x* `offset` *a* `| metric2 multiplier` *y* `offset` *b*`)`	Calculate a metric based on the current values of `metric` and `metric2`. This policy action overrides the current value of the metric attribute with the result of the expression ((x * metric) + a) + ((y * metric2) + b) where `metric` and `metric2` are the current input values. Metric multipliers are limited in range to eight significant digits.
`metric (igp | minimum-igp)` *site-offset*	(BGP only) Change the metric (MED) value by the specified negative or positive offset. This action is useful only in an external BGP (EBGP) export policy.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description

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

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description

	protocol bgp;
	rib vrf-customer1.inet.0;
	community customer;
	}
	then {
	next-hop next-table vrf-customer2.inet.0;
	}
	}
	
	If you specify `reject`, the next-hop address is replaced by a reject next hop.

`origin` *value*	(BGP only) Set the BGP origin attribute to one of the following values: • `igp`—Path information originated within the local AS. • `egp`—Path information originated in another AS. • `incomplete`—Path information learned by some other means.
`p2mp-lsp-root`	Set the ingress root node for a multipoint LDP (M-LDP)-based point-to-multipoint label-switched path (LSP). For more information, see Example: Configuring Multipoint LDP In-Band Signaling for Point-to-Multipoint LSPs.
`preference` *preference* `preference2` *preference*	Set the preference value. You can specify a primary preference value (`preference`) and a secondary preference value (`preference2`). The preference value can be a number in the range from 0 through 4,294,967,295 (2^{32} – 1). A lower number indicates a more preferred route. When you use an import policy to set the value of `preference2` to the highest allowed value of 4,294,967,295, Junos OS resets this value to -1. If you set `preference2` to a number greater than (2^{31} – 1), it is reset to a negative value. To specify even finer-grained preference values, see the `color` and `color2` actions in this table. If you set the preference with the `preference` action, the new preference remains associated with the route. The new preference is internal to the Junos OS and is not transitive.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`preference (add | subtract) number preference2 (add | subtract) number`	Change the preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
`priority (low | medium | high)`	(OSPF import only) Specify a priority for prefixes included in an OSPF import policy. Prefixes learned through OSPF are installed in the routing table based on the priority assigned to the prefixes. Prefixes assigned a priority of `high` are installed first, while prefixes assigned a priority of `low` are installed last. **NOTE**: An OSPF import policy can only be used to set priority or to filter OSPF external routes. If an OSPF import policy is applied that results in a `reject` terminating action for a nonexternal route, then the `reject` action is ignored and the route is accepted anyway.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`source-class` *`source-class-name`*	Maintain packet counts for a route passing through your network, based on the source address. You can do the following: • Configure group source prefixes by configuring a routing policy. • Apply that routing policy to the forwarding table with the corresponding source class. • Enable packet counting on one or more interfaces by including the `source-class-usage` *`interface-name`* statement at the `[edit interfaces` *`logical-unit-number`* `unit family inet accounting]` hierarchy level. Also, follow the `source-class-usage` statement with the `input` or `output` statement to define the inbound and outbound interfaces on which traffic monitored for source-class usage (SCU) is arriving and departing (or define one interface for both). The complete syntax is `[edit interfaces` *`interface-name`* `unit family inet accounting source-class-usage (input | output | input output)` *`unit-number`*`]`. • View the output by using one of the following commands: `show interfaces` *`interface-name`* `source-class` *`source-class-name`*, `show interfaces` *`interface-name`* `extensive`, or `show interfaces` *`interface-name`* `statistics` (see the CLI Explorer). • To configure a packet count based on the destination address, use the `destination-class` statement described in this table. • For a detailed source-class usage example configuration, see the "Example: Grouping Source and Destination Prefixes into a Forwarding Class" on page 671. **NOTE**: When configuring policy action statements, you can configure only one source class for each matching route. In other words, more than one source class cannot be applied to the same route.
`ssm-source [` *`addresses`* `];`	Specify one or more IPv4 or IPv6 source addresses for the source-specific multicast (SSM) policy
`ssm-source [` *`addresses`* `];`	Specify one or more IPv4 or IPv6 source addresses for the source-specific multicast (SSM) policy.

Table 10: Actions That Manipulate Route Characteristics *(Continued)*

Action	Description
`tag` *tag* `tag2` *tag*	Set the tag value. You can specify two tag strings: `tag` (for the first string) and `tag2` (a second string). These values are local to the router. • For OSPF routes the `tag` action sets the 32-bit tag field in OSPF external link-state advertisement (LSA) packets. • For IS-IS routes, the `tag` action sets the 32-bit flag in the IS-IS IP prefix type length values (TLV). • For RIPv2 routes, the `tag` action sets the route-tag community. The `tag2` option is not supported.
`tag (add | subtract)` *number* `tag2 (add | subtract)` *number*	Change the tag value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
`validation-state`	When BGP origin validation is configured, set the validation state of a route prefix to valid, invalid, or unknown. The route validation database contains route origin authorization (ROA) records that map route prefixes to expected originating autonomous systems (ASs). This prevents the accidental advertisement of invalid routes. See Understanding Origin Validation for BGP.

Configuring the Default Action in Routing Policies

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

```
[edit]
policy-options {
    policy-statement policy-name {
```

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

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

Example: Configuring the Default Action in a Routing Policy

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

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

```
        }
        term spoke {
            from interface [ ge-2/1/0.1 ge-2/1/0.2 ];
            then {
                community add test-01-spoke;
                default-action accept;
            }
        }
        term management {
            from protocol direct;
            then {
                community add management;
                default-action accept;
            }
        }
    }
}
```

Configuring a Final Action in Routing Policies

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

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

```
            }
        }
        then action;
    }
}
```

Logging Matches to a Routing Policy Term

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

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

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

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

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

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

Configuring Separate Actions for Routes in Route Lists

If you specify route lists in the `from` statement, for each route in the list, you can specify an action to take on that individual route directly, without including a `then` statement. For more information, see "Understanding Route Filters for Use in Routing Policy Match Conditions" on page 303.

RELATED DOCUMENTATION

Route Filter Match Conditions | 72

Routing Policy Match Conditions | 58

Summary of 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.

All policies 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.

An action defines what the router does with the route when the route matches all the match conditions in the `from` and `to` statements for a particular term. If a term does not have `from` and `to` statements, all routes are considered to match and the actions apply to all routes.

Each term can have one or more of the following types of actions. The actions are configured under the `then` statement.

- 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

If you do not specify an action, one of the following results occurs:

- The next term in the routing policy, if one exists, is evaluated.
- If the routing policy has no more terms, the next routing policy, if one exists, is evaluated.
- If there are no more terms or routing policies, the accept or reject action specified by the default policy is executed.

Table 11 on page 93 summarizes the routing policy actions.

Table 11: Summary of Key Routing Policy Actions

Action	Description
Flow Control Actions	These actions control the flow of routing information into and out of the routing table.
`accept`	Accepts the route and propagates it. After a route is accepted, no other terms in the routing policy and no other routing policies are evaluated.
`reject`	Rejects the route and does not propagate it. After a route is rejected, no other terms in the routing policy and no other routing policies are evaluated.
`next term`	Skips to and evaluates the next term in the same routing policy. Any `accept` or `reject` action specified in the `then` statement is ignored. Any actions specified in the `then` statement that manipulate route characteristics are applied to the route.
`next policy`	Skips to and evaluates the next routing policy. Any `accept` or `reject` action specified in the `then` statement is ignored. Any actions specified in the `then` statement that manipulate route characteristics are applied to the route.
Route Manipulation Actions	These actions manipulate the route characteristics.

Table 11: Summary of Key Routing Policy Actions *(Continued)*

Action	Description
`as-path-prepend` *as-path*	Appends one or more AS numbers at the beginning of the AS path. If you are specifying more than one AS number, include 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 appended AS numbers are placed within a confederation sequence. Otherwise, the appended AS numbers are placed with a nonconfederation sequence.
`as-path-expand last-as count` *n*	Extracts the last AS number in the existing AS path and appends that AS number to the beginning of the AS path *n* times. Replace *n* with a number from 1 through 32. 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 appended AS numbers are placed within a confederation sequence. Otherwise, the appended AS numbers are placed with a nonconfederation sequence.
`class` *class-name*	Applies the specified class-of-service (CoS) parameters to routes installed into the routing table.
`color` *preference* `color2` *preference*	Sets the preference value to the specified value. The `color` and `color2` preference values can be a number from `0` through `4,294,967,295` (2^{32} - 1). A lower number indicates a more preferred route.
`damping` *name*	Applies the specified route-damping parameters to the route. These parameters override BGP's default damping parameters. This action is useful only in import policies.
`local-preference` *value*	Sets the BGP local preference attribute. The preference can be a number from `0` through `4,294,967,295` (2^{32} - 1).

Table 11: Summary of Key Routing Policy Actions *(Continued)*

Action	Description
`metric` *metric* `metric2` *metric* `metric3` *metric* `metric4` *metric*	Sets 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`. For BGP routes, `metric` corresponds to the MED, and `metric2` corresponds to the IGP metric if the BGP next hop loops through another router.
`next-hop` *address*	Sets the next hop. If you specify *address* as `self`, the next-hop address is replaced by one of the local router's addresses. The advertising protocol determines which address to use.

RELATED DOCUMENTATION

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Example: Configuring a Routing Policy to Advertise the Best External Route to Internal Peers

IN THIS SECTION

- Requirements | 97
- Overview | 97
- Configuration | 99
- Verification | 103

The BGP protocol specification, as defined in RFC 1771, specifies that a BGP peer shall advertise to its internal peers the higher preference external path, even if this path is not the overall best (in other

words, even if the best path is an internal path). In practice, deployed BGP implementations do not follow this rule. The reasons for deviating from the specification are as follows:

- Minimizing the amount of advertised information. BGP scales according to the number of available paths.
- Avoiding routing and forwarding loops.

There are, however, several scenarios in which the behavior, specified in RFC 1771, of advertising the best external route might be beneficial. Limiting path information is not always desirable as path diversity might help reduce restoration times. Advertising the best external path can also address internal BGP (IBGP) route oscillation issues as described in RFC 3345, *Border Gateway Protocol (BGP) Persistent Route Oscillation Condition*.

The `advertise-external` statement modifies the behavior of a BGP speaker to advertise the best external path to IBGP peers, even when the best overall path is an internal path.

> **NOTE**: The `advertise-external` statement is supported at both the group and neighbor level. If you configure the statement at the neighbor level, you must configure it for all neighbors in a group. Otherwise, the group is automatically split into different groups.

The `conditional` option limits the behavior of the `advertise-external` setting, such that the external route is advertised only if the route selection process reaches the point where the multiple exit discriminator (MED) metric is evaluated. Thus, an external route is not advertised if it has, for instance, an AS path that is worse (longer) than that of the active path. The `conditional` option restricts external path advertisement to when the best external path and the active path are equal until the MED step of the route selection process. Note that the criteria used for selecting the best external path is the same whether or not the `conditional` option is configured.

Junos OS also provides support for configuring a BGP export policy that matches the state of an advertised route. You can match either active or inactive routes, as follows:

```
policy-options {
    policy-statement name{
        from state (active|inactive);
    }
}
```

This qualifier only matches when used in the context of an export policy. When a route is being advertised by a protocol that can advertise inactive routes (such as BGP), `state inactive` matches routes advertised as a result of the `advertise-inactive` and `advertise-external` statements.

For example, the following configuration can be used as a BGP export policy toward internal peers to mark routes advertised due to the `advertise-external` setting with a user-defined community. That community can be later used by the receiving routers to filter out such routes from the forwarding table. Such a mechanism can be used to address concerns that advertising paths not used for forwarding by the sender might lead to forwarding loops.

```
user@host# show policy-options
policy-statement mark-inactive {
    term inactive {
        from state inactive;
        then {
            community set comm-inactive;
        }
    }
    term default {
        from protocol bgp;
        then accept;
    }
    then reject;
}
community comm-inactive members 65536:65284;
```

Requirements

Junos OS 9.3 or later is required.

Overview

IN THIS SECTION

- Topology | 98

This example shows three routing devices. Device R2 has an external BGP (EBGP) connection to Device R1. Device R2 has an IBGP connection to Device R3.

Device R1 advertises 172.16.6.0/24. Device R2 does not set the local preference in an import policy for Device R1's routes, and thus 172.16.6.0/24 has the default local preference of 100.

Device R3 advertises 172.16.6.0/24 with a local preference of 200.

When the advertise-external statement is not configured on Device R2, 172.16.6.0/24 is not advertised by Device R2 toward Device R3.

When the advertise-external statement is configured on Device R2 on the session toward Device R3, 172.16.6.0/24 is advertised by Device R2 toward Device R3.

When advertise-external conditional is configured on Device R2 on the session toward Device R3, 172.16.6.0/24 is not advertised by Device R2 toward Device R3. If you remove the then local-preference 200 setting on Device R3 and add the path-selection as-path-ignore setting on Device R2 (thus making the path selection criteria equal until the MED step of the route selection process), 172.16.6.0/24 is advertised by Device R2 toward Device R3.

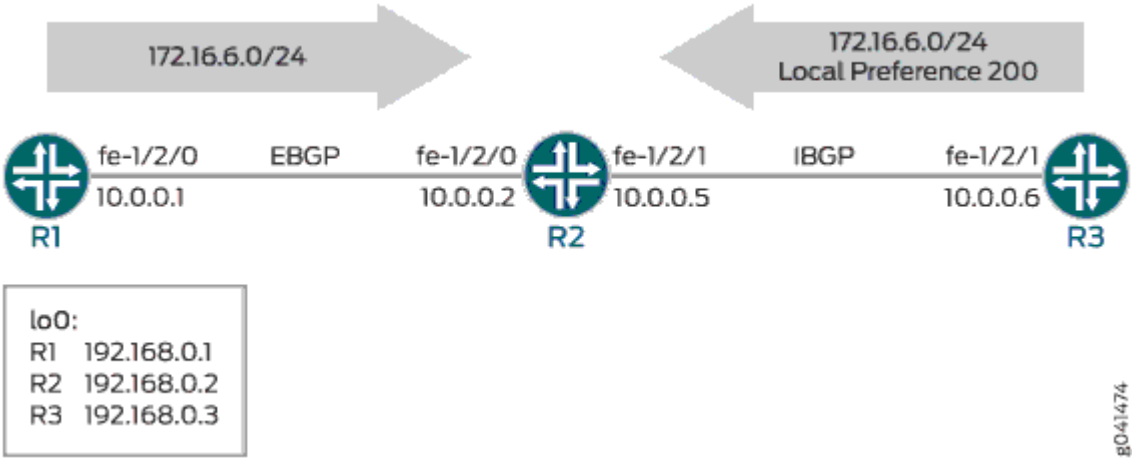
NOTE: To configure the advertise-external statement on a route reflector, you must disable intracluster reflection with the no-client-reflect statement, and the client cluster must be fully meshed to prevent the sending of redundant route advertisements.

When a routing device is configured as a route reflector for a cluster, a route advertised by the route reflector is considered internal if it is received from an internal peer with the same cluster identifier or if both peers have no cluster identifier configured. A route received from an internal peer that belongs to another cluster, that is, with a different cluster identifier, is considered external.

Topology

Figure 9 on page 98 shows the sample network.

Figure 9: BGP Topology for advertise-external



"CLI Quick Configuration" on page 99 shows the configuration for all of the devices in Figure 9 on page 98.

The section "No Link Title" on page 101 describes the steps on Device R2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 99
- Procedure | 100

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 description to-R2
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 from route-filter 172.16.6.0/24 exact
set policy-options policy-statement send-static term 1 then accept
set policy-options policy-statement send-static term 2 then reject
set routing-options static route 172.16.6.0/24 reject
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 description to-R1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
```

```
set interfaces fe-1/2/1 unit 0 description to-R3
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.5/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 100
set protocols bgp group ext neighbor 10.0.0.1
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int advertise-external
set protocols bgp group int neighbor 192.168.0.3
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 200
```

Device R3

```
set interfaces fe-1/2/0 unit 6 family inet address 10.0.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int export send-static
set protocols bgp group int neighbor 192.168.0.2
set protocols ospf area 0.0.0.0 interface fe-1/2/0.6
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then local-preference 200
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 172.16.6.0/24 reject
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.5
set routing-options autonomous-system 200
```

Procedure

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 description to-R1
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set fe-1/2/1 unit 0 description to-R3
user@R2# set fe-1/2/1 unit 0 family inet address 10.0.0.5/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure OSPF or another interior gateway protocol (IGP).

```
[edit protocols ospf area 0.0.0.0]
user@R2# set interface fe-1/2/1.0
user@R2# set interface lo0.0 passive
```

3. Configure the EBGP connection to Device R1.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set peer-as 100
user@R2# set neighbor 10.0.0.1
```

4. Configure the IBGP connection to Device R3.

```
[edit protocols bgp group int]
user@R2# set type internal
user@R2# set local-address 192.168.0.2
user@R2# set neighbor 192.168.0.3
```

5. Add the `advertise-external` statement to the IBGP group peering session.

```
[edit protocols bgp group int]
user@R2# set advertise-external
```

6. Configure the autonomous system (AS) number and the router ID.

```
[edit routing-options ]
user@R2# set router-id 192.168.0.2
user@R2# set autonomous-system 200
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0{
        description to-R1;
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        description to-R3;
        family inet {
            address 10.0.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
```

```
    group ext {
        type external;
        peer-as 100;
        neighbor 10.0.0.1;
    }
    group int {
        type internal;
        local-address 192.168.0.2;
        advertise-external;
        neighbor 192.168.0.3;
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
```

```
user@R2# show routing-options
router-id 192.168.0.2;
autonomous-system 200;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the BGP Active Path | **104**
- Verifying the External Route Advertisement | **104**
- Verifying the Route on Device R3 | **105**
- Experimenting with the conditional Option | **105**

Confirm that the configuration is working properly.

Verifying the BGP Active Path

Purpose

On Device R2, make sure that the 172.16.6.0/24 prefix is in the routing table and has the expected active path.

Action

```
user@R2> show route 172.16.6

inet.0: 8 destinations, 9 routes (8 active, 1 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.6.0/24      *[BGP/170] 00:00:07, localpref 200, from 192.168.0.3
                      AS path: I, validation-state: unverified
                     > to 10.0.0.6 via fe-1/2/1.0
                     [BGP/170] 03:23:03, localpref 100
                      AS path: 100 I, validation-state: unverified
                     > to 10.0.0.1 via fe-1/2/0.0
```

Meaning

Device R2 receives the 172.16.6.0/24 route from both Device R1 and Device R3. The route from Device R3 is the active path, as designated by the asterisk (*). The active path has the highest local preference. Even if the local preferences of the two routes were equal, the route from Device R3 would remain active because it has the shortest AS path.

Verifying the External Route Advertisement

Purpose

On Device R2, make sure that the 172.16.6.0/24 route is advertised toward Device R3.

Action

```
user@R2> show route advertising-protocol bgp 192.168.0.3

inet.0: 8 destinations, 9 routes (8 active, 1 holddown, 0 hidden)
```

```
Prefix                  Nexthop              MED     Lclpref    AS path
172.16.6.0/24           10.0.0.1                     100        100 I
```

Meaning

Device R2 is advertising the 172.16.6.0/24 route toward Device R3.

Verifying the Route on Device R3

Purpose

Make sure that the 172.16.6.0/24 prefix is in Device R3's routing table.

Action

```
user@R3> show route 172.16.6.0/24

inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.6.0/24      *[Static/5] 03:34:14
                      Reject
                    [BGP/170] 06:34:43, localpref 100, from 192.168.0.2
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/0.6
```

Meaning

Device R3 has the static route and the BGP route for 172.16.6.0/24.

Note that the BGP route is hidden on Device R3 if the route is not reachable or if the next hop cannot be resolved. To fulfill this requirement, this example includes a static default route on Device R3 (`static route 0.0.0.0/0 next-hop 10.0.0.5`).

Experimenting with the conditional Option

Purpose

See how the `conditional` option works in the context of the BGP path selection algorithm.

Action

1. On Device R2, add the `conditional` option.

```
[edit  protocols bgp group int]
user@R2# set advertise-external conditional
user@R2# commit
```

2. On Device R2, check to see if the 172.16.6.0/24 route is advertised toward Device R3.

```
user@R2> show route advertising-protocol bgp 192.168.0.3
```

   As expected, the route is no longer advertised. You might need to wait a few seconds to see this result.

3. On Device R3, deactivate the `then local-preference` policy action.

```
[edit policy-options policy-statement send-static term 1]
user@R3# deactivate logical-systems R3 then local-preference
user@R3# commit
```

4. On Device R2, ensure that the local preferences of the two paths are equal.

```
user@R2> show route 172.16.6.0/24

inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.6.0/24      *[BGP/170] 08:02:59, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
                    [BGP/170] 00:07:51, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/1.0
```

5. On Device R2, add the `as-path-ignore` statement.

```
[edit protocols bgp]
user@R2# set path-selection as-path-ignore
user@R2# commit
```

6. On Device R2, check to see if the 172.16.6.0/24 route is advertised toward Device R3.

```
user@R2> show route advertising-protocol bgp 192.168.0.3

inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.6.0/24           10.0.0.1                     100        100 I
```

As expected, the route is now advertised because the AS path length is ignored and because the local preferences are equal.

RELATED DOCUMENTATION

Example: Configuring BGP to Advertise Inactive Routes

Understanding BGP Path Selection

Example: Configuring BGP to Advertise Inactive Routes

IN THIS SECTION

- Requirements | **109**
- Overview | **109**
- Configuration | **110**
- Verification | **114**

By default, BGP readvertises only active routes. To have the routing table export to BGP the best route learned by BGP even if Junos OS did not select it to be an active route, include the `advertise-inactive` statement:

```
advertise-inactive;
```

In Junos OS, BGP advertises BGP routes that are installed or active, which are routes selected as the best based on the BGP path selection rules. The `advertise-inactive` statement allows nonactive BGP routes to be advertised to other peers.

NOTE: If the routing table has two BGP routes where one is active and the other is inactive, the `advertise-inactive` statement does not advertise the inactive BGP prefix. This statement does not advertise an inactive BGP route in the presence of another active BGP route. However, if the active route is a static route, the `advertise-inactive` statement advertises the inactive BGP route.

NOTE: The `advertise-inactive` statement does not help to advertise the inactive route from the VRF when the router is configured as a route reflector.

Junos OS also provides support for configuring a BGP export policy that matches the state of an advertised route. You can match either active or inactive routes, as follows:

```
policy-options {
    policy-statement name{
        from state (active|inactive);
    }
}
```

This qualifier only matches when used in the context of an export policy. When a route is being advertised by a protocol that can advertise inactive routes (such as BGP), `state inactive` matches routes advertised as a result of the `advertise-inactive` (or `advertise-external`) statement.

For example, the following configuration can be used as a BGP export policy to mark routes advertised due to the `advertise-inactive` setting with a user-defined community. That community can be later used by the receiving routers to filter out such routes from the forwarding table. Such a mechanism can be

used to address concerns that advertising paths not used for forwarding by the sender might lead to forwarding loops.

```
user@host# show policy-options
policy-statement mark-inactive {
    term inactive {
        from state inactive;
        then {
            community set comm-inactive;
        }
    }
    term default {
        from protocol bgp;
        then accept;
    }
    then reject;
}
community comm-inactive members 65536:65284;
```

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | **110**

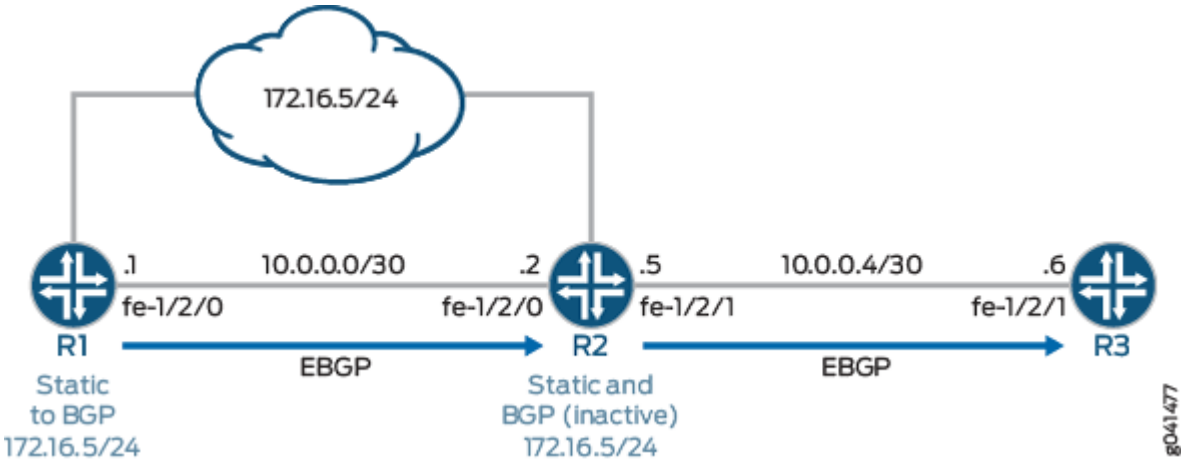
In this example, Device R2 has two external BGP (EBGP) peers, Device R1 and Device R3.

Device R1 has a static route to 172.16.5/24. Likewise, Device R2 also has a static route to 172.16.5/24. Through BGP, Device R1 sends information about its static route to Device R2. Device R2 now has information about 172.16.5/24 from two sources—its own static route and the BGP-learned route received from Device R1. Static routes are preferred over BGP-learned routes, so the BGP route is inactive on Device R2. Normally Device R2 would send the BGP-learned information to Device R3, but Device R2 does not do this because the BGP route is inactive. Device R3, therefore, has no information about 172.16.5/24 unless you enable the `advertise-inactive` command on Device R2, which causes Device R2 to send the BGP-learned to Device R3.

Topology

Figure 10 on page 110 shows the sample network.

Figure 10: BGP Topology for advertise-inactive



"CLI Quick Configuration" on page 110 shows the configuration for all of the devices in Figure 10 on page 110.

The section "No Link Title" on page 112 describes the steps on Device R2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 110
- Procedure | 112

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group to_R2 type external
set protocols bgp group to_R2 export send-static
set protocols bgp group to_R2 neighbor 10.0.0.2 peer-as 200
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 172.16.5.0/24 discard
set routing-options static route 172.16.5.0/24 install
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.5/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group to_R1 type external
set protocols bgp group to_R1 neighbor 10.0.0.1 peer-as 100
set protocols bgp group to_R3 type external
set protocols bgp group to_R3 advertise-inactive
set protocols bgp group to_R3 neighbor 10.0.0.6 peer-as 300
set routing-options static route 172.16.5.0/24 discard
set routing-options static route 172.16.5.0/24 install
set routing-options autonomous-system 200
```

Device R3

```
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.6/30
set interfaces fe-1/2/0 unit 9 family inet address 10.0.0.9/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.5
set routing-options autonomous-system 300
```

Procedure

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set fe-1/2/1 unit 0 family inet address 10.0.0.5/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the EBGP connection to Device R1.

```
[edit protocols bgp group to_R1]
user@R2# set type external
user@R2# set neighbor 10.0.0.1 peer-as 100
```

3. Configure the EBGP connection to Device R3.

```
[edit protocols bgp group to_R3]
user@R2# set type external
user@R2# set neighbor 10.0.0.6 peer-as 300
```

4. Add the `advertise-inactive` statement to the EBGP group peering session with Device R3.

```
[edit protocols bgp group to_R3]
user@R2# set advertise-inactive
```

5. Configure the static route to the 172.16.5.0/24 network.

```
[edit routing-options static]
user@R2# set route 172.16.5.0/24 discard
user@R2# set route 172.16.5.0/24 install
```

6. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set autonomous-system 200
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 10.0.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
```

```
    }
}
```

```
user@R2# show protocols
bgp {
    group to_R1 {
        type external;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
    }
    group to_R3 {
        type external;
        advertise-inactive;
        neighbor 10.0.0.6 {
            peer-as 300;
        }
    }
}
```

```
user@R2# show routing-options
static {
    route 172.16.5.0/24 {
        discard;
        install;
    }
}
autonomous-system 200;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the BGP Active Path | **115**
- Verifying the External Route Advertisement | **115**
- Verifying the Route on Device R3 | **116**

- Experimenting with the advertise-inactive Statement | 117

Confirm that the configuration is working properly.

Verifying the BGP Active Path

Purpose

On Device R2, make sure that the 172.16.5.0/24 prefix is in the routing table and has the expected active path.

Action

```
user@R2> show route 172.16.5

inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.5.0/24      *[Static/5] 21:24:38
                      Discard
                    [BGP/170] 21:21:41, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
```

Meaning

Device R2 receives the 172.16.5.0/24 route from both Device R1 and from its own statically configured route. The static route is the active path, as designated by the asterisk (*). The static route path has the lowest route preference (5) as compared to the BGP preference (170). Therefore, the static route becomes active.

Verifying the External Route Advertisement

Purpose

On Device R2, make sure that the 172.16.5.0/24 route is advertised toward Device R3.

Action

```
user@R2> show route advertising-protocol bgp 10.0.0.6


inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  172.16.5.0/24           Self                                    100 I
```

Meaning

Device R2 is advertising the 172.16.5.0/24 route toward Device R3

Verifying the Route on Device R3

Purpose

Make sure that the 172.16.6.0/24 prefix is in Device R3's routing table.

Action

```
user@R3> show route 172.16.5.0/24

inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.5.0/24      *[BGP/170] 00:01:19, localpref 100
                      AS path: 200 100 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/1.0
```

Meaning

Device R3 has the BGP-learned route for 172.16.5.0/24.

Experimenting with the advertise-inactive Statement

Purpose

See what happens when the `advertise-inactive` statement is removed from the BGP configuration on Device R2.

Action

1. On Device R2, deactivate the `advertise-inactive` statement.

```
[edit  protocols bgp group to_R3]
user@R2# deactivate advertise-inactive
user@R2# commit
```

2. On Device R2, check to see if the 172.16.5.0/24 route is advertised toward Device R3.

```
user@R2> show route advertising-protocol bgp 10.0.0.6
```

   As expected, the route is no longer advertised.

3. On Device R3, ensure that the 172.16.5/24 route is absent from the routing table.

```
user@R3> show route 172.16.5/24
```

Meaning

Device R1 advertises route 172.16.5/24 to Device R2, but Device R2 has a manually configured static route for this prefix. Static routes are preferred over BGP routes, so Device R2 installs the BGP route as an inactive route. Because the BGP route is not active, Device R2 does not readvertise the BGP route to Device R3. This is the default behavior in Junos OS. If you add the `advertise-inactive` statement to the BGP configuration on Device R2, Device R2 readvertises nonactive routes.

RELATED DOCUMENTATION

> *Example: Configuring a Routing Policy to Advertise the Best External Route to Internal Peers*

Example: Using Routing Policy to Set a Preference Value for BGP Routes

IN THIS SECTION

- Requirements | **118**
- Overview | **118**
- Configuration | **120**
- Verification | **125**

This example shows how to use routing policy to set the preference for routes learned from BGP. Routing information can be learned from multiple sources. To break ties among equally specific routes learned from multiple sources, each source has a preference value. Routes that are learned through explicit administrative action, such as static routes, are preferred over routes learned from a routing protocol, such as BGP or OSPF. This concept is called *administrative distance* by some vendors.

Requirements

No special configuration beyond device initialization is required before you configure this example.

Overview

IN THIS SECTION

- Topology | **119**

Routing information can be learned from multiple sources, such as through static configuration, BGP, or an interior gateway protocol (IGP). When Junos OS determines a route's preference to become the active route, it selects the route with the lowest preference as the active route and installs this route into the forwarding table. By default, the routing software assigns a preference of 170 to routes that originated from BGP. Of all the routing protocols, BGP has the highest default preference value, which means that routes learned by BGP are the least likely to become the active route.

Some vendors have a preference (distance) of 20 for external BGP (EBGP) and a distance of 200 for internal BGP (IGBP). Junos OS uses the same value (170) for both EBGP and IBGP. However, this difference between vendors has no operational impact because Junos OS always prefers EBGP routes over IBGP routes.

Another area in which vendors differ is in regard to IGP distance compared to BGP distance. For example, some vendors assign a distance of 110 to OSPF routes. This is higher than the EBGP distance of 20, and results in the selection of an EBGP route over an equivalent OSPF route. In the same scenario, Junos OS chooses the OSPF route, because of the default preference 10 for an internal OSPF route and 150 for an external OSPF route, which are both lower than the 170 preference assigned to all BGP routes.

This example shows a routing policy that matches routes from specific next hops and sets a preference. If a route does not match the first term, it is evaluated by the second term.

Topology

In the sample network, Device R1 and Device R3 have EBGP sessions with Device R2.

On Device R2, an import policy takes the following actions:

- For routes received through BGP from next-hop 10.0.0.1 (Device R1), set the route preference to 10.
- For routes received through BGP from next-hop 10.1.0.2 (Device R3), set the route preference to 15.

Figure 11 on page 119 shows the sample network.

Figure 11: BGP Preference Value Topology

"CLI Quick Configuration" on page 120 shows the configuration for all of the devices in Figure 11 on page 119.

The section "No Link Title" on page 121 describes the steps on Device R2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 120
- Procedure | 121

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext import set-preference
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement set-preference term term1 from protocol bgp
```

```
set policy-options policy-statement set-preference term term1 from next-hop 10.0.0.1
set policy-options policy-statement set-preference term term1 then preference 10
set policy-options policy-statement set-preference term term2 from protocol bgp
set policy-options policy-statement set-preference term term2 from next-hop 10.1.0.2
set policy-options policy-statement set-preference term term2 then preference 15
set routing-options autonomous-system 200
```

Device R3

```
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.1.0.1
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 300
```

Procedure

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set fe-1/2/1 unit 0 family inet address 10.1.0.1/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the local autonomous system.

```
[edit routing-options]
user@R2# set autonomous-system 200
```

3. Configure the routing policy that sends direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

4. Configure the routing policy that changes the preference of received routes.

```
[edit policy-options policy-statement set-preference]
user@R2# set term term1 from protocol bgp
user@R2# set term term1 from next-hop 10.0.0.1
user@R2# set term term1 then preference 10
user@R2# set term term2 from protocol bgp
user@R2# set term term2 from next-hop 10.1.0.2
user@R2# set term term2 then preference 15
```

5. Configure the external peering with Device R2.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set export send-direct
user@R2# set neighbor 10.0.0.1 peer-as 100
user@R2# set neighbor 10.1.0.2 peer-as 300
```

6. Apply the `set-preference` policy as an import policy.

   This affects Device R2's routing table and has no impact on Device R1 and Device R3.

```
[edit protocols bgp group ext]
user@R2# set import set-preference
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 10.1.0.1/30;
        }
    }
}
lo0 {
    unit 0{
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group ext {
        type external;
        import set-preference;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
        neighbor 10.1.0.2 {
            peer-as 300;
        }
```

```
    }
}
```

```
user@R2# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
policy-statement set-preference {
    term term1 {
        from {
            protocol bgp;
            next-hop 10.0.0.1;
        }
        then {
            preference 10;
        }
    }
    term term2 {
        from {
            protocol bgp;
            next-hop 10.1.0.2;
        }
        then {
            preference 15;
        }
    }
}
```

```
user@R2# show routing-options
autonomous-system 200;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Preference | 125

Confirm that the configuration is working properly.

Verifying the Preference

Purpose

Make sure that the routing tables on Device R1 and Device R2 reflect the fact that Device R1 is using the configured EBGP preference of 8, and Device R2 is using the default EBGP preference of 170.

Action

From operational mode, enter the `show route protocols bgp` command.

```
user@R2> show route protocols bgp
inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/30         [BGP/10] 04:42:23, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
10.1.0.0/30         [BGP/15] 04:42:23, localpref 100
                      AS path: 300 I, validation-state: unverified
                    > to 10.1.0.2 via fe-1/2/1.0
192.168.0.1/32     *[BGP/10] 04:42:23, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
192.168.0.3/32     *[BGP/15] 04:42:23, localpref 100
                      AS path: 300 I, validation-state: unverified
                    > to 10.1.0.2 via fe-1/2/1.0
```

Meaning

The output shows that on Device R2, the preference values have been changed to 15 for routes learned from Device R3, and the preference values have been changed to 10 for routes learned from Device R1.

RELATED DOCUMENTATION

- Route Preferences Overview
- Understanding External BGP Peering Sessions

Example: Enabling BGP Route Advertisements

IN THIS SECTION

- Requirements | 127
- Overview | 127
- Configuration | 128
- Verification | 134

Junos OS does not advertise the routes learned from one EBGP peer back to the same external BGP (EBGP) peer. In addition, the software does not advertise those routes back to any EBGP peers that are in the same autonomous system (AS) as the originating peer, regardless of the routing instance. You can modify this behavior by including the `advertise-peer-as` statement in the configuration.

If you include the `advertise-peer-as` statement in the configuration, BGP advertises the route regardless of this check.

To restore the default behavior, include the `no-advertise-peer-as` statement in the configuration:

```
no-advertise-peer-as;
```

The route suppression default behavior is disabled if the `as-override` statement is included in the configuration. If you include both the `as-override` and `no-advertise-peer-as` statements in the configuration, the `no-advertise-peer-as` statement is ignored.

Requirements

No special configuration beyond device initialization is required before you configure this example.

NOTE: This example was updated and re-validated on Junos release 21.2R1.

Overview

IN THIS SECTION

- Topology | 127

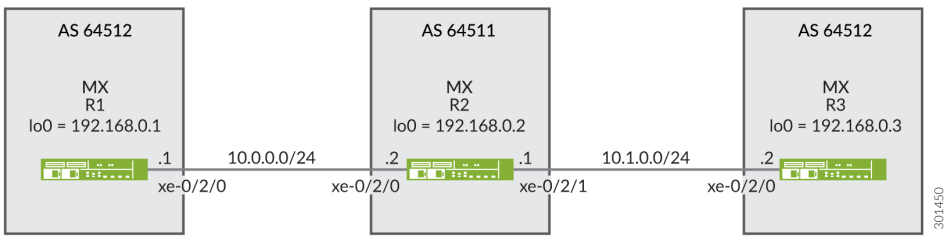
This example shows three routing devices with external BGP (EBGP) connections. Device R2 has an EBGP connection to Device R1 and another EBGP connection to Device R3. Although separated by Device R2 which is in AS 64511, Device R1 and Device R3 are in the same AS (AS 64512). Device R1 and Device R3 advertise into BGP direct routes to their own loopback interface addresses.

Device R2 receives these loopback interface routes, and the `advertise peer-as` statement allows Device R2 to advertise them. Specifically, Device R1 sends the 192.168.0.1 route to Device R2, and because Device R2 has the `advertise peer-as` configured, Device R2 can send the 192.168.0.1 route to Device R3. Likewise, Device R3 sends the 192.168.0.3 route to Device R2, and `advertise peer-as` enables Device R2 to forward the route to Device R1.

To enable Device R1 and Device R3 to accept routes that contain their own AS number in the AS path, the `loops 2` statement is required on Device R1 and Device R3.

Topology

Figure 12: BGP Topology for advertise-peer-as



Configuration

IN THIS SECTION

- CLI Quick Configuration | 128
- Procedure | 129

CLI Quick Configuration

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

Device R1

```
set interfaces xe-0/2/0 description R1-to-R2
set interfaces xe-0/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp family inet unicast loops 2
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 64512
```

Device R2

```
set interfaces xe-0/2/0 description R2-to-R1
set interfaces xe-0/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces xe-0/2/1 description R2-to-R3
set interfaces xe-0/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext advertise-peer-as
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 64512
```

```
set protocols bgp group ext neighbor 10.1.0.2 peer-as 64512
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 64511
```

Device R3

```
set interfaces xe-0/2/0 description R3-to-R2
set interfaces xe-0/2/0 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp family inet unicast loops 2
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.1.0.1
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 64512
```

Procedure

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set xe-0/2/0 description R1-to-R2
user@R1# set xe-0/2/0 unit 0 family inet address 10.0.0.1/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure BGP.

```
[edit protocols bgp group ext]
user@R1# set type external
```

```
user@R1# set peer-as 64511
user@R1# set neighbor 10.0.0.2
```

3. Prevent routes from Device R3 from being hidden on Device R1 by including the `loops 2` statement.

   The `loops 2` statement means that the local device's own AS number can appear in the AS path up to one time without causing the route to be hidden. The route is hidden if the local device's AS number is detected in the path two or more times.

```
[edit protocols bgp family inet unicast]
user@R1# set loops 2
```

4. Configure the routing policy that sends direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R1# set from protocol direct
user@R1# set then accept
```

5. Apply the export policy to the BGP peering session with Device R2.

```
[edit protocols bgp group ext]
user@R1# set export send-direct
```

6. Configure the autonomous system (AS) number.

```
[edit routing-options ]
user@R1# set autonomous-system 64512
```

Step-by-Step Procedure

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R2# set xe-0/2/0 description R2-to-R1
user@R2# set xe-0/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set xe-0/2/1 description R2-to-R3
```

```
user@R2# set xe-0/2/1 unit 0 family inet address 10.1.0.1/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure BGP.

```
[edit  protocols bgp group ext]
user@R2# set type external
user@R2# set neighbor 10.0.0.1 peer-as 64512
user@R2# set neighbor 10.1.0.2 peer-as 64512
```

3. Configure Device R2 to advertise routes learned from one EBGP peer to another EBGP peer in the same AS.

   In other words, advertise to Device R1 routes learned from Device R3 (and the reverse), even though Device R1 and Device R3 are in the same AS.

```
[edit  protocols bgp group ext]
user@R2# set advertise-peer-as
```

4. Configure a routing policy that sends direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

5. Apply the export policy.

```
[edit protocols bgp group ext]
user@R2# set export send-direct
```

6. Configure the AS number.

```
[edit routing-options]
user@R2# set autonomous-system 64511
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R1

```
user@R1# show interfaces
xe-0/2/0 {
    description R1-to-R2;
    unit 0 {
        family inet {
            address 10.0.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    family inet {
        unicast {
            loops 2;
        }
    }
    group ext {
        type external;
        export send-direct;
        peer-as 64511;
        neighbor 10.0.0.2;
```

```
    }
}
```

```
user@R1# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R1# show routing-options
autonomous-system 64512;
```

Device R2

```
user@R2# show interfaces
xe-0/2/0 {
    description R2-to-R1;
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
xe-0/2/1 {
    description R2-to-R3;
    unit 0 {
        family inet {
            address 10.1.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
```

```
    }
}
```

```
user@R2# show protocols
bgp {
    group ext {
        type external;
        advertise-peer-as;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 64512;
        }
        neighbor 10.1.0.2 {
            peer-as 64512;
        }
    }
}
```

```
user@R2# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R2# show routing-options
autonomous-system 64511;
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the BGP Routes | 135

Confirm that the configuration is working properly.

Verifying the BGP Routes

Purpose

Make sure that the routing tables on Device R1 and Device R3 contain the expected routes.

Action

1. On Device R2, deactivate the `advertise-peer-as` statement in the BGP configuration.

```
[edit  protocols bgp group ext]
user@R2# deactivate advertise-peer-as
user@R2# commit
```

2. On Device R3, deactivate the `loops` statement in the BGP configuration.

```
[edit  protocols bgp family inet unicast ]
user@R3# deactivate unicast loops
user@R3# commit
```

3. On Device R1, check to see what routes are advertised to Device R2.

```
user@R1> show route advertising-protocol bgp 10.0.0.2
inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.0/30             Self                                    I
* 192.168.0.1/32          Self                                    I
```

4. On Device R2, check to see what routes are received from Device R1.

```
user@R2> show route receive-protocol bgp 10.0.0.1
inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  10.0.0.0/30             10.0.0.1                                64512 I
* 192.168.0.1/32          10.0.0.1                                64512 I
```

5. On Device R2, check to see what routes are advertised to Device R3.

```
user@R2> show route advertising-protocol bgp 10.1.0.2
inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.0/30             Self                                    I
* 10.1.0.0/30             Self                                    I
* 192.168.0.2/32          Self                                    I
```

6. On Device R2, activate the `advertise-peer-as` statement in the BGP configuration.

```
[edit  protocols bgp group ext]
user@R2# activate advertise-peer-as
user@R2# commit
```

7. On Device R2, recheck the routes that are advertised to Device R3.

```
user@R2> show route advertising-protocol bgp 10.1.0.2
inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.0/30             Self                                    I
* 10.1.0.0/30             Self                                    I
* 192.168.0.1/32          Self             64512 I
* 192.168.0.2/32          Self                                    I
* 192.168.0.3/32          10.1.0.2         64512 I
```

8. On Device R3, check the routes that are received from Device R2.

```
user@R3> show route receive-protocol bgp 10.1.0.1
inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.0/30             10.1.0.1                                64511 I
  10.1.0.0/30             10.1.0.1                                64511 I
* 192.168.0.2/32          10.1.0.1                                64511 I
```

9. On Device R3, activate the loops statement in the BGP configuration.

```
[edit  protocols bgp family inet unicast ]
user@R3# activate unicast loops
user@R3# commit
```

10. On Device R3, recheck the routes that are received from Device R2.

```
user@R3> show route receive-protocol bgp 10.1.0.1
inet.0: 6 destinations, 8 routes (6 active, 0 holddown, 1 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.0/30             10.1.0.1                                64511 I
  10.1.0.0/30             10.1.0.1                                64511 I
* 192.168.0.1/32          10.1.0.1                                64511 64512 I
* 192.168.0.2/32          10.1.0.1                                64511 I
```

Meaning

First the advertise-peer-as statement and the loops statement are deactivated so that the default behavior can be examined. Device R1 sends to Device R2 a route to Device R1's loopback interface address, 192.168.0.1/32. Device R2 does not advertise this route to Device R3. After activating the advertise-peer-as statement, Device R2 does advertise the 192.168.0.1/32 route to Device R3. Device R3 does not accept this route until after the loops statement is activated.

RELATED DOCUMENTATION

Example: Configuring a Layer 3 VPN with Route Reflection and AS Override

Example: Rejecting Known Invalid Routes

IN THIS SECTION

- Requirements | 138
- Overview | 138

Configuration | 138
Verification | 140

This example shows how to create route-based match conditions for a routing policy.

Requirements

Before you begin, be sure your router interfaces and protocols are correctly configured.

Overview

IN THIS SECTION

Topology | 138

In this example, you create a policy called rejectpolicy1 that rejects routes with a mask of /8 and greater (/8, /9, /10, and so on) that have the first 8 bits set to 0. This policy also accepts routes less than 8 bits in length by creating a mask of 0/0 up to /7.

Topology

Configuration

IN THIS SECTION

Procedure | 139

Procedure

CLI Quick Configuration

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

```
set policy-options policy-statement rejectpolicy1 term rejectterm1 from route-filter 0.0.0.0/0
upto /7 accept
set policy-options policy-statement rejectpolicy1 term rejectterm1 from route-filter 0.0.0.0/8
orlonger reject
set policy-options policy-statement test term 1 from protocol direct
```

Step-by-Step Procedure

To create a policy that rejects known invalid routes:

1. Create the routing policy.

```
[edit]
user@host# edit policy-options policy-statement rejectpolicy1
```

2. Create the policy term.

```
[edit policy-options policy-statement rejectpolicy1]
user@host# edit term rejectterm1
```

3. Create a mask that specifies which routes to accept.

```
[edit policy-options policy-statement rejectpolicy1 term rejectterm1]
user@host# set from route-filter 0/0 upto /7 accept
```

4. Create a mask that specifies which routes to reject.

```
[edit policy-options policy-statement rejectpolicy1 term rejectterm1]
user@host# set from route-filter 0/8 orlonger reject
```

Results

Confirm your configuration by entering the **show policy-options** command from configuration mode. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host# show policy-options
policy-statement rejectpolicy1 {
    term rejectterm1 {
        from {
            route-filter 0.0.0.0/0 upto /7 accept;
            route-filter 0.0.0.0/8 orlonger reject;
        }
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying the Route-Based Match Conditions | **140**

To confirm that the configuration is working properly, perform these tasks:

Verifying the Route-Based Match Conditions

Purpose

Verify that the policy and term are configured on the device with the appropriate route-based match conditions.

Action

From operational mode, enter the **show policy-options** command.

RELATED DOCUMENTATION

Route Filter Match Conditions | 72

Example: Grouping Source and Destination Prefixes into a Forwarding Class | 671

Example: Using Routing Policy in an ISP Network

IN THIS SECTION

- Requirements | 141
- Overview | 141
- Set Commands for All Devices in the Topology | 144
- Configuring Device Customer-1 | 152
- Configuring Device Customer-2 | 155
- Configuring Devices ISP-1 and ISP-2 | 159
- Configuring Device ISP-3 | 166
- Configuring Device Exchange-2 | 173
- Configuring Device Private-Peer-2 | 176
- Verification | 182

This example is a case study in how routing policies might be used in a typical Internet service provider (ISP) network.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 142

In this network example, the ISP's AS number is 64510. The ISP has two transit peers (AS 64514 and AS 64515) to which it connects at an exchange point. The ISP is also connected to two private peers (AS 64513 and AS 64516) with which it exchanges specific customer routes. The ISP has two customers (AS 64511 and AS 64512).

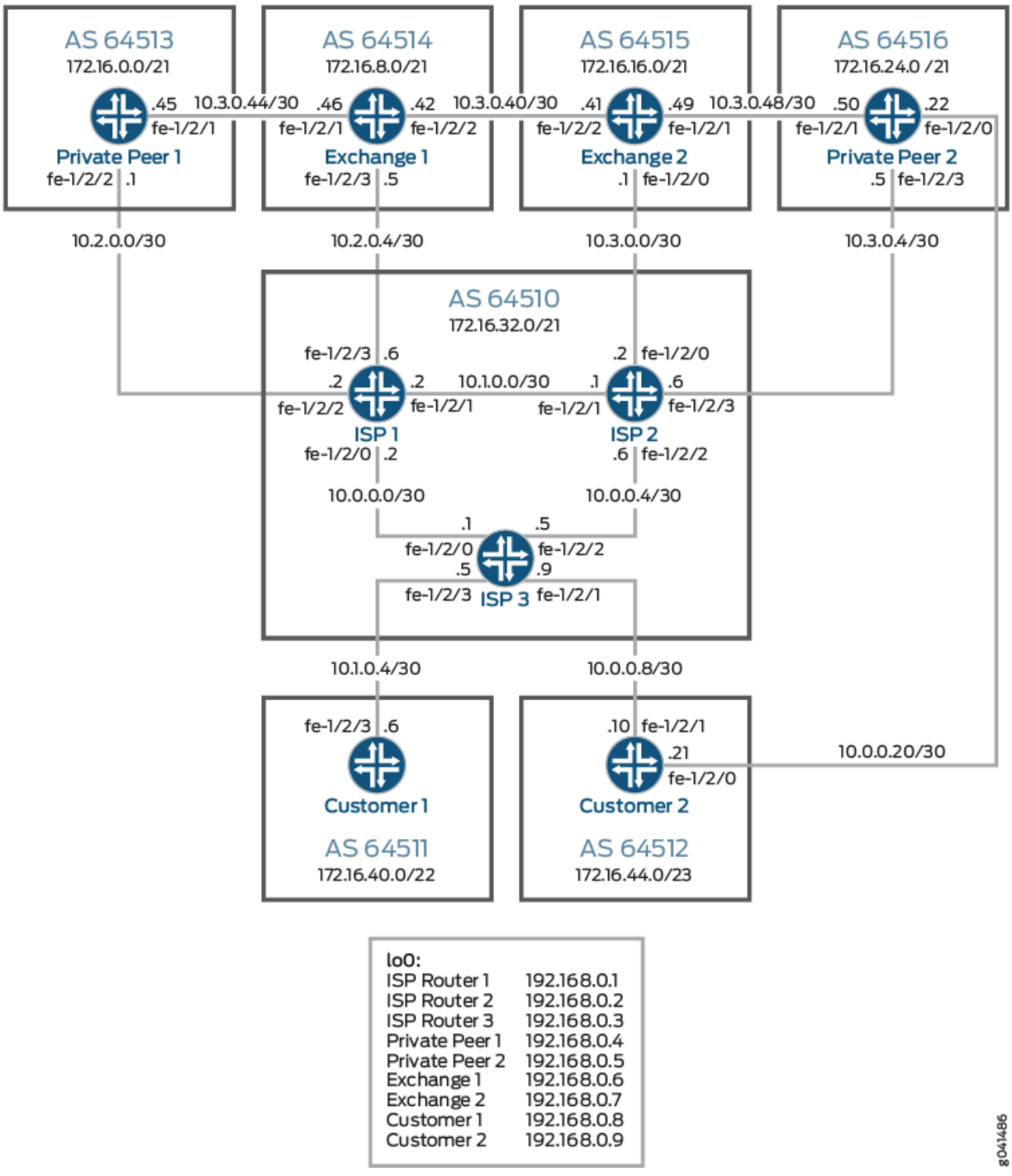
The ISP policies are configured in an outbound direction. That is, the example focuses on the routes that the ISP announces to its peers and customers, and includes the following:

1. The ISP has been assigned AS 64510 and the routing space of 172.16.32.0/21. With the exception of the two customer networks, all other customer routes are simulated with static routes.
2. The exchange 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 administrators want all data to the private peers to use the direct links. 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, each customer has a different set of requirements. Customer-1 requires a singe default route. Customer-2 requires specific routes.

Topology

Figure 13 on page 143 shows the sample network.

Figure 13: ISP Network Example



Set Commands for All Devices in the Topology

IN THIS SECTION

- CLI Quick Configuration | 144

CLI Quick Configuration

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

Device Customer-1

```
set interfaces fe-1/2/3 unit 0 description to_ISP-3
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.8/32
set protocols bgp group ext type external
set protocols bgp group ext export send-statics
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.1.0.5
set policy-options policy-statement send-statics term static-routes from protocol static
set policy-options policy-statement send-statics term static-routes then accept
set routing-options static route 172.16.40.0/25 reject
set routing-options static route 172.16.40.128/25 reject
set routing-options static route 172.16.41.0/25 reject
set routing-options static route 172.16.41.128/25 reject
set routing-options autonomous-system 64511
```

Device Customer-2

```
set interfaces fe-1/2/1 unit 0 description to_ISP-3
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.10/30
set interfaces fe-1/2/0 unit 0 description to-Private-Peer-2
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.21/30
set interfaces lo0 unit 0 family inet address 192.168.0.9/32
set protocols bgp group ext type external
set protocols bgp group ext import inbound-routes
set protocols bgp group ext export outbound-routes
```

```
set protocols bgp group ext neighbor 10.0.0.9 peer-as 64510
set protocols bgp group ext neighbor 10.0.0.22 peer-as 64516
set policy-options policy-statement inbound-routes term AS64510-primary from protocol bgp
set policy-options policy-statement inbound-routes term AS64510-primary from as-path AS64510-
routes
set policy-options policy-statement inbound-routes term AS64510-primary then local-preference 200
set policy-options policy-statement inbound-routes term AS64510-primary then accept
set policy-options policy-statement inbound-routes term AS64516-backup from protocol bgp
set policy-options policy-statement inbound-routes term AS64516-backup from as-path AS64516-
routes
set policy-options policy-statement inbound-routes term AS64516-backup then local-preference 50
set policy-options policy-statement inbound-routes term AS64516-backup then accept
set policy-options policy-statement outbound-routes term statics from protocol static
set policy-options policy-statement outbound-routes term statics then accept
set policy-options policy-statement outbound-routes term internal-bgp-routes from protocol bgp
set policy-options policy-statement outbound-routes term internal-bgp-routes from as-path my-own-
routes
set policy-options policy-statement outbound-routes term internal-bgp-routes then accept
set policy-options policy-statement outbound-routes term no-transit then reject
set policy-options as-path my-own-routes "()"
set policy-options as-path AS64510-routes "64510 .*"
set policy-options as-path AS64516-routes "64516 .*"
set routing-options static route 172.16.44.0/26 reject
set routing-options static route 172.16.44.64/26 reject
set routing-options static route 172.16.44.128/26 reject
set routing-options static route 172.16.44.192/26 reject
set routing-options autonomous-system 64512
```

Device ISP-1

```
set interfaces fe-1/2/0 unit 0 description to_ISP-3
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 description to_ISP-2
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces fe-1/2/2 unit 0 description to_Private-Peer-1
set interfaces fe-1/2/2 unit 0 family inet address 10.2.0.2/30
set interfaces fe-1/2/3 unit 0 description to_Exchange-1
set interfaces fe-1/2/3 unit 0 family inet address 10.2.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.1
set protocols bgp group int export internal-peers
```

```
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group to_64513 type external
set protocols bgp group to_64513 export private-peer
set protocols bgp group to_64513 peer-as 64513
set protocols bgp group to_64513 neighbor 10.2.0.1
set protocols bgp group to_64514 type external
set protocols bgp group to_64514 export exchange-peer
set protocols bgp group to_64514 peer-as 64514
set protocols bgp group to_64514 neighbor 10.2.0.5
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement exchange-peer term AS64510-Aggregate from protocol aggregate
set policy-options policy-statement exchange-peer term AS64510-Aggregate from route-filter
172.16.32.0/21 exact
set policy-options policy-statement exchange-peer term AS64510-Aggregate then accept
set policy-options policy-statement exchange-peer term Customer-2-Aggregate from protocol
aggregate
set policy-options policy-statement exchange-peer term Customer-2-Aggregate from route-filter
172.16.40.0/22 exact
set policy-options policy-statement exchange-peer term Customer-2-Aggregate then accept
set policy-options policy-statement exchange-peer term reject-all-other-routes then reject
set policy-options policy-statement internal-peers term statics from protocol static
set policy-options policy-statement internal-peers term statics then accept
set policy-options policy-statement internal-peers term next-hop-self then next-hop self
set policy-options policy-statement private-peer term statics from protocol static
set policy-options policy-statement private-peer term statics then accept
set policy-options policy-statement private-peer term isp-and-customer-routes from protocol bgp
set policy-options policy-statement private-peer term isp-and-customer-routes from route-filter
172.16.32.0/21 orlonger
set policy-options policy-statement private-peer term isp-and-customer-routes then accept
set policy-options policy-statement private-peer term reject-all then reject
set routing-options static route 172.16.32.0/24 reject
set routing-options static route 172.16.33.0/24 reject
set routing-options aggregate route 172.16.32.0/21
set routing-options aggregate route 172.16.40.0/22
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device ISP-2

```
set interfaces fe-1/2/1 unit 0 description to_ISP-1
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces fe-1/2/2 unit 0 description to_ISP-3
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.6/30
set interfaces fe-1/2/3 unit 0 description to_Private-Peer-2
set interfaces fe-1/2/3 unit 0 family inet address 10.3.0.6/30
set interfaces fe-1/2/0 unit 0 description to_Exchange-2
set interfaces fe-1/2/0 unit 0 family inet address 10.3.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int export internal-peers
set protocols bgp group int neighbor 192.168.0.1
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group AS-64516 type external
set protocols bgp group AS-64516 export private-peer
set protocols bgp group AS-64516 peer-as 64516
set protocols bgp group AS-64516 neighbor 10.3.0.5
set protocols bgp group AS-64515 type external
set protocols bgp group AS-64515 export exchange-peer
set protocols bgp group AS-64515 peer-as 64515
set protocols bgp group AS-64515 neighbor 10.3.0.1
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement exchange-peer term AS64510-Aggregate from protocol aggregate
set policy-options policy-statement exchange-peer term AS64510-Aggregate from route-filter
172.16.32.0/21 exact
set policy-options policy-statement exchange-peer term AS64510-Aggregate then accept
set policy-options policy-statement exchange-peer term Customer-2-Aggregate from protocol
aggregate
set policy-options policy-statement exchange-peer term Customer-2-Aggregate from route-filter
172.16.44.0/23 exact
set policy-options policy-statement exchange-peer term Customer-2-Aggregate then accept
set policy-options policy-statement exchange-peer term reject-all-other-routes then reject
set policy-options policy-statement internal-peers term statics from protocol static
set policy-options policy-statement internal-peers term statics then accept
set policy-options policy-statement internal-peers term next-hop-self then next-hop self
set policy-options policy-statement private-peer term statics from protocol static
set policy-options policy-statement private-peer term statics then accept
```

```
set policy-options policy-statement private-peer term isp-and-customer-routes from protocol bgp
set policy-options policy-statement private-peer term isp-and-customer-routes from route-filter
172.16.32.0/21 orlonger
set policy-options policy-statement private-peer term isp-and-customer-routes then accept
set policy-options policy-statement private-peer term reject-all then reject
set routing-options static route 172.16.34.0/24 reject
set routing-options static route 172.16.35.0/24 reject
set routing-options aggregate route 172.16.44.0/23
set routing-options aggregate route 172.16.32.0/21
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64510
```

Device ISP-3

```
set interfaces fe-1/2/0 unit 0 description to_ISP-1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/2 unit 0 description to_ISP-2
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.5/30
set interfaces fe-1/2/3 unit 0 description to_Customer-1
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.5/30
set interfaces fe-1/2/1 unit 0 description to_Customer-2
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.9/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int export internal-peers
set protocols bgp group int neighbor 192.168.0.1
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group to_64511 type external
set protocols bgp group to_64511 export customer-1-peer
set protocols bgp group to_64511 neighbor 10.1.0.6 peer-as 64511
set protocols bgp group to_64512 type external
set protocols bgp group to_64512 export customer-2-peer
set protocols bgp group to_64512 neighbor 10.0.0.10 peer-as 64512
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement customer-1-peer term defaut-route from route-filter
0.0.0.0/0 exact
set policy-options policy-statement customer-1-peer term defaut-route then accept
set policy-options policy-statement customer-1-peer term reject-all-other-routes then reject
set policy-options policy-statement customer-2-peer term statics from protocol static
```

```
set policy-options policy-statement customer-2-peer term statics then accept
set policy-options policy-statement customer-2-peer term isp-and-customer-routes from protocol
bgp
set policy-options policy-statement customer-2-peer term isp-and-customer-routes from route-
filter 172.16.32.0/21 orlonger
set policy-options policy-statement customer-2-peer term isp-and-customer-routes then accept
set policy-options policy-statement customer-2-peer term default-route from route-filter
0.0.0.0/0 exact
set policy-options policy-statement customer-2-peer term default-route then accept
set policy-options policy-statement customer-2-peer term reject-all-other-routes then reject
set policy-options policy-statement if-upstream-routes-exist term only-certain-contributing-
routes from route-filter 172.16.8.0/21 exact
set policy-options policy-statement if-upstream-routes-exist term only-certain-contributing-
routes then accept
set policy-options policy-statement if-upstream-routes-exist term reject-all-other-routes then
reject
set policy-options policy-statement internal-peers term statics from protocol static
set policy-options policy-statement internal-peers term statics then accept
set policy-options policy-statement internal-peers term next then next-hop self
set routing-options static route 172.16.36.0/24 reject
set routing-options static route 172.16.37.0/24 reject
set routing-options static route 172.16.38.0/24 reject
set routing-options static route 172.16.39.0/24 reject
set routing-options generate route 0.0.0.0/0 policy if-upstream-routes-exist
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64510
```

Device Exchange-1

```
set interfaces fe-1/2/3 unit 0 description to_ISP-1
set interfaces fe-1/2/3 unit 0 family inet address 10.2.0.5/30
set interfaces fe-1/2/2 unit 0 description to_Exchange-2
set interfaces fe-1/2/2 unit 0 family inet address 10.3.0.42/30
set interfaces fe-1/2/1 unit 0 description to_Private-Peer-1
set interfaces fe-1/2/1 unit 0 family inet address 10.3.0.45/30
set interfaces lo0 unit 0 family inet address 192.168.0.6/32
set protocols bgp group ext type external
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.2.0.6
set protocols bgp group ext neighbor 10.3.0.41 peer-as 64515
set policy-options policy-statement send-static from protocol static
```

```
set policy-options policy-statement send-static then accept
set routing-options static route 172.16.8.0/21 reject
set routing-options autonomous-system 64514
```

Device Exchange-2

```
set interfaces fe-1/2/0 unit 0 description to_ISP-2
set interfaces fe-1/2/0 unit 0 family inet address 10.3.0.1/30
set interfaces fe-1/2/2 unit 0 description to_Exchange-1
set interfaces fe-1/2/2 unit 0 family inet address 10.3.0.41/30
set interfaces fe-1/2/1 unit 0 description to_Private-Peer-2
set interfaces fe-1/2/1 unit 0 family inet address 10.3.0.49/30
set interfaces lo0 unit 0 family inet address 192.168.0.7/32
set protocols bgp group ext type external
set protocols bgp group ext export outbound-routes
set protocols bgp group ext neighbor 10.3.0.2 peer-as 64510
set protocols bgp group ext neighbor 10.3.0.50 peer-as 64516
set protocols bgp group ext neighbor 10.3.0.42 peer-as 64514
set policy-options policy-statement outbound-routes term statics from protocol static
set policy-options policy-statement outbound-routes term statics then accept
set routing-options autonomous-system 64515
set routing-options static route 172.16.16.0/21 reject
```

Device Private-Peer-1

```
set interfaces fe-1/2/2 unit 0 description to_ISP-1
set interfaces fe-1/2/2 unit 0 family inet address 10.2.0.1/30
set interfaces fe-1/2/1 unit 0 description to_Exchange-1
set interfaces fe-1/2/1 unit 0 family inet address 10.3.0.46/30
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.2.0.2
set routing-options autonomous-system 64513
```

Device Private-Peer-2

```
set interfaces fe-1/2/3 unit 0 description to_ISP-2
set interfaces fe-1/2/3 unit 0 family inet address 10.3.0.5/30
set interfaces fe-1/2/0 unit 0 description to_Customer-1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.22/30
```

```
set interfaces fe-1/2/1 unit 0 description to_Exchange-2
set interfaces fe-1/2/1 unit 0 family inet address 10.3.0.50/30
set interfaces lo0 unit 0 family inet address 192.168.0.5/32
set protocols bgp group ext type external
set protocols bgp group ext export outbound-routes
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.3.0.6
set protocols bgp group to-64512 type external
set protocols bgp group to-64512 peer-as 64512
set protocols bgp group to-64512 neighbor 10.0.0.21
set protocols bgp group to-64512 export internal-routes
set protocols bgp group to-64515 type external
set protocols bgp group to-64515 export outbound-routes
set protocols bgp group to-64515 peer-as 64515
set protocols bgp group to-64515 neighbor 10.3.0.49
set policy-options policy-statement if-upstream-routes-exist term as-64515-routes from route-
filter 172.16.16.0/21 exact
set policy-options policy-statement if-upstream-routes-exist term as-64515-routes then accept
set policy-options policy-statement if-upstream-routes-exist term reject-all-other-routes then
reject
set policy-options policy-statement internal-routes term statics from protocol static
set policy-options policy-statement internal-routes term statics then accept
set policy-options policy-statement internal-routes term default-route from route-filter
0.0.0.0/0 exact
set policy-options policy-statement internal-routes term default-route then accept
set policy-options policy-statement internal-routes term reject-all-other-routes then reject
set policy-options policy-statement outbound-routes term statics from protocol static
set policy-options policy-statement outbound-routes term statics then accept
set policy-options policy-statement outbound-routes term allowed-bgp-routes from as-path my-own-
routes
set policy-options policy-statement outbound-routes term allowed-bgp-routes from as-path AS64512-
routes
set policy-options policy-statement outbound-routes term allowed-bgp-routes then accept
set policy-options policy-statement outbound-routes term no-transit then reject
set policy-options as-path my-own-routes "()"
set policy-options as-path AS64512-routes 64512
set routing-options static route 172.16.24.0/25 reject
set routing-options static route 172.16.24.128/25 reject
set routing-options static route 172.16.25.0/26 reject
set routing-options static route 172.16.25.64/26 reject
set routing-options generate route 0.0.0.0/0 policy if-upstream-routes-exist
set routing-options autonomous-system 64516
```

Configuring Device Customer-1

IN THIS SECTION

- Procedure | 152

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

Device Customer-1 has multiple static routes configured to simulate customer routes. These routes are sent to the ISP.

To configure Device Customer-1:

1. Configure the device interfaces.

```
[edit interfaces]
user@Customer-1# set fe-1/2/3 unit 0 description to_ISP-3
user@Customer-1# set fe-1/2/3 unit 0 family inet address 10.1.0.6/30
user@Customer-1# set lo0 unit 0 family inet address 192.168.0.8/32
```

2. Configure the static routes.

```
[edit routing-options static]
user@Customer-1# set route 172.16.40.0/25 reject
user@Customer-1# set route 172.16.40.128/25 reject
user@Customer-1# set route 172.16.41.0/25 reject
user@Customer-1# set route 172.16.41.128/25 reject
```

3. Configure the policy to send static routes.

```
[edit policy-options policy-statement send-statics term static-routes]
user@Customer-1# set from protocol static
user@Customer-1# set then accept
```

4. Configure the external BGP (EBGP) connection to the ISP.

```
[edit protocols bgp group ext]
user@Customer-1# set type external
user@Customer-1# set export send-statics
user@Customer-1# set peer-as 64510
user@Customer-1# set neighbor 10.1.0.5
```

5. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@Customer-1# set autonomous-system 64511
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@Customer-1# show interfaces
fe-1/2/1 {
    unit 0 {
        description to_ISP-3;
        family inet {
            address 10.1.0.6/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.8/32;
```

```
        }
    }
}
```

```
user@Customer-1# show protocols
bgp {
    group ext {
        type external;
        export send-statics;
        peer-as 64510;
        neighbor 10.1.0.5;
    }
}
```

```
user@Customer-1# show policy-options
policy-statement send-statics {
    term static-routes {
        from protocol static;
        then accept;
    }
}
```

```
user@Customer-1# show routing-options
static {
    route 172.16.40.0/25 reject;
    route 172.16.40.128/25 reject;
    route 172.16.41.0/25 reject;
    route 172.16.41.128/25 reject;
}
autonomous-system 64511;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device Customer-2

IN THIS SECTION

- Procedure | 155

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

Device Customer-2 has two static routes configured to simulate customer routes. These routes are sent to the ISP. 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 64516. Customer-2 wants to use the ISP for transit service to the Internet, and has requested a default route from the ISP.

To configure Device Customer-2:

1. Configure the device interfaces.

```
[edit interfaces]
user@Customer-2# set fe-1/2/1 unit 0 description to_ISP-3
user@Customer-2# set fe-1/2/1 unit 0 family inet address 10.0.0.10/30
user@Customer-2# set fe-1/2/0 unit 0 description to-Private-Peer-2
user@Customer-2# set fe-1/2/0 unit 0 family inet address 10.0.0.21/30
user@Customer-2# set lo0 unit 0 family inet address 192.168.0.9/32
```

2. Configure the static routes.

```
[edit routing-options static]
user@Customer-2# set route 172.16.44.0/26 reject
user@Customer-2# set route 172.16.44.64/26 reject
user@Customer-2# set route 172.16.44.128/26 reject
user@Customer-2# set route 172.16.44.192/26 reject
```

3. Configure the import routing policy.

The route with the highest local preference value is preferred. Routes from the ISP are preferred over the same routes from Device Private-Peer-2

```
[edit policy-options policy-statement inbound-routes]
user@Customer-2# set term AS64510-primary from protocol bgp
user@Customer-2# set term AS64510-primary from as-path AS64510-routes
user@Customer-2# set term AS64510-primary then local-preference 200
user@Customer-2# set term AS64510-primary then accept
[edit policy-options policy-statement inbound-routes]
user@Customer-2# set term AS64516-backup from protocol bgp
user@Customer-2# set term AS64516-backup from as-path AS64516-routes
user@Customer-2# set term AS64516-backup then local-preference 50
user@Customer-2# set term AS64516-backup then accept
[edit policy-options]
user@Customer-2# set as-path AS64510-routes "64510 .*"
user@Customer-2# set as-path AS64516-routes "64516 .*"
```

4. Configure the export routing policy.

```
[edit policy-options policy-statement outbound-routes]
user@Customer-2# set term statics from protocol static
user@Customer-2# set term statics then accept
user@Customer-2# set term internal-bgp-routes from protocol bgp
user@Customer-2# set term internal-bgp-routes from as-path my-own-routes
user@Customer-2# set term internal-bgp-routes then accept
user@Customer-2# set term no-transit then reject
[edit policy-options]
user@Customer-2# set as-path my-own-routes "()"
```

5. Configure the external BGP (EBGP) connection to the ISP and to Device Private-Peer-2.

```
[edit protocols bgp group ext]
user@Customer-2# set type external
user@Customer-2# set import inbound-routes
user@Customer-2# set export outbound-routes
user@Customer-2# set neighbor 10.0.0.9 peer-as 64510
user@Customer-2# set neighbor 10.0.0.22 peer-as 64516
```

6. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@Customer-2# set autonomous-system 64512
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@Customer-2# show interfaces
fe-1/2/1 {
    unit 0 {
        description to_ISP-3;
        family inet {
            address 10.0.0.10/30;
        }
    }
}
fe-1/2/0 {
    unit 0 {
        description to-Private-Peer-2;
        family inet {
            address 10.0.0.21/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.9/32;
        }
    }
}
```

```
user@Customer-2# show protocols
bgp {
    group ext {
```

```
        type external;
        import inbound-routes;
        export outbound-routes;
        neighbor 10.0.0.9 {
            peer-as 64510;
        }
        neighbor 10.0.0.22 {
            peer-as 64516;
        }
    }
}
```

```
user@Customer-2# show policy-options
policy-statement inbound-routes {
    term AS64510-primary {
        from {
            protocol bgp;
            as-path AS64510-routes;
        }
        then {
            local-preference 200;
            accept;
        }
    }
    term AS64516-backup {
        from {
            protocol bgp;
            as-path AS64516-routes;
        }
        then {
            local-preference 50;
            accept;
        }
    }
}
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;
    }
}
as-path my-own-routes "()";
as-path AS64510-routes "64510 .*";
as-path AS64516-routes "64516 .*";
```

```
user@Customer-2# show routing-options
static {
    route 172.16.44.0/26 reject;
    route 172.16.44.64/26 reject;
    route 172.16.44.128/26 reject;
    route 172.16.44.192/26 reject;
}
autonomous-system 64512;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Devices ISP-1 and ISP-2

IN THIS SECTION

- Procedure | 159

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

Device ISP-1 and Device ISP-2 each have two policies configured: The `private-peer` policy and the `exchange-peer` policy. Because of their similar configurations, this example shows the step-by-step configuration only for Device ISP-2.

On Device ISP-2, the private-peer policy sends the ISP customer routes to Device Private-Peer-2. The policy accepts all local static routes (local Device ISP-2 customers) and all BGP routes in the 172.16.32.0/21 range (advertised by other ISP routers). These two policy terms represent the ISP customer routes. The final policy 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 Device Private-Peer-2 for two reasons:

- The peer already maintains a connection to Device Exchange-2 in our example, so the routes are redundant.
- The private peer wants customer routes only. The `private-peer` policy accomplishes this goal. The `exchange-peer` policy sends routes to Device Exchange-2.

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

- The aggregate route that represents the AS 64510 routing space of 172.16.32.0/21. This route is configured as an aggregate route locally and is advertised by the `exchange-peer` policy.
- The address space assigned to Customer-2, 172.16.44.0/23. This smaller aggregate route needs to be sent to Device Exchange-2 because the customer is also attached to the AS 64516 peer (Device Private-Peer-2).

Sending these two routes to Device Exchange-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 /23 network while the ISP maintained only its /21 aggregate, all traffic destined for the customer would transit AS 64516 only. Because the customer also wants routes from the ISP, the 172.16.44.0/23 route is announced by Device ISP-2. Like the larger aggregate route, the 172.16.44.0/23 route is configured locally and is advertised by the exchange-peer policy. The final term in that policy rejects all routes, including the specific customer networks of the ISP, the customer routes from Device Private-Peer-1, the customer routes from Device Private-Peer-2, and the routing table from Device Exchange-1. In essence, this final term prevents the ISP from performing transit services for the Internet at large.

To configure Device ISP-2:

1. Configure the device interfaces.

```
[edit interfaces]
user@ISP-2# set fe-1/2/1 unit 0 description to_ISP-1
user@ISP-2# set fe-1/2/1 unit 0 family inet address 10.1.0.1/30
user@ISP-2# set fe-1/2/2 unit 0 description to_ISP-3
user@ISP-2# set fe-1/2/2 unit 0 family inet address 10.0.0.6/30
```

```
user@ISP-2# set fe-1/2/3 unit 0 description to_Private-Peer-2
user@ISP-2# set fe-1/2/3 unit 0 family inet address 10.3.0.6/30
user@ISP-2# set fe-1/2/0 unit 0 description to_Exchange-2
user@ISP-2# set fe-1/2/0 unit 0 family inet address 10.3.0.2/30
user@ISP-2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the interior gateway protocol (IGP).

```
[edit protocols ospf area 0.0.0.0]
user@ISP-2# set interface fe-1/2/2.0
user@ISP-2# set interface fe-1/2/1.0
user@ISP-2# set interface lo0.0 passive
```

3. Configure the static and aggregate routes.

```
[edit routing-options static]
user@ISP-2# set route 172.16.34.0/24 reject
user@ISP-2# set route 172.16.35.0/24 reject
[edit routing-options aggregate]
user@ISP-2# set route 172.16.44.0/23
user@ISP-2# set route 172.16.32.0/21
```

4. Configure the routing policies for the exchange peers.

```
[edit policy-options policy-statement exchange-peer]
user@ISP-2# set term AS64510-Aggregate from protocol aggregate
user@ISP-2# set term AS64510-Aggregate from route-filter 172.16.32.0/21 exact
user@ISP-2# set term AS64510-Aggregate then accept
user@ISP-2# set term Customer-2-Aggregate from protocol aggregate
user@ISP-2# set term Customer-2-Aggregate from route-filter 172.16.44.0/23 exact
user@ISP-2# set term Customer-2-Aggregate then accept
user@ISP-2# set term reject-all-other-routes then reject
```

5. Configure the routing policies for the internal peers.

```
[edit policy-options policy-statement internal-peers]
user@ISP-2# set term statics from protocol static
```

```
user@ISP-2# set term statics then accept
user@ISP-2# set term next-hop-self then next-hop self
```

6. Configure the routing policies for the private peer.

```
[edit policy-options policy-statement private-peer]
user@ISP-2# set term statics from protocol static
user@ISP-2# set term statics then accept
user@ISP-2# set term isp-and-customer-routes from protocol bgp
user@ISP-2# set term isp-and-customer-routes from route-filter 172.16.32.0/21 orlonger
user@ISP-2# set term isp-and-customer-routes then accept
user@ISP-2# set term reject-all then reject
```

7. Configure the internal BGP (IBGP) connections to the other ISP devices.

```
[edit protocols bgp group int]
user@ISP-2# set type internal
user@ISP-2# set local-address 192.168.0.2
user@ISP-2# set export internal-peers
user@ISP-2# set neighbor 192.168.0.1
user@ISP-2# set neighbor 192.168.0.3
```

8. Configure the EBGP connections to the exchange peer and the private peer.

```
[edit protocols bgp group AS-64516]
user@ISP-2# set type external
user@ISP-2# set export private-peer
user@ISP-2# set peer-as 64516
user@ISP-2# set neighbor 10.3.0.5
[edit protocols bgp group AS-64515]
user@ISP-2# set type external
user@ISP-2# set export exchange-peer
user@ISP-2# set peer-as 64515
user@ISP-2# set neighbor 10.3.0.1
```

9. Configure the autonomous system (AS) number and the router ID.

```
[edit routing-options]
user@ISP-2# set router-id 192.168.0.2
user@ISP-2# set autonomous-system 64510
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@ISP-2# show interfaces
fe-1/2/0 {
    unit 0{
        description to_Exchange-2;
        family inet {
            address 10.3.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0{
        description to_ISP-1;
        family inet {
            address 10.1.0.1/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        description to_ISP-3;
        family inet {
            address 10.0.0.6/30;
        }
    }
}
fe-1/2/3 {
    unit 0 {
        description to_Private-Peer-2;
```

```
        family inet {
            address 10.3.0.6/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@ISP-2# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.2;
        export internal-peers;
        neighbor 192.168.0.1;
        neighbor 192.168.0.3;
    }
    group AS-64516 {
        type external;
        export private-peer;
        peer-as 64516;
        neighbor 10.3.0.5;
    }
    group AS-64515 {
        type external;
        export exchange-peer;
        peer-as 64515;
        neighbor 10.3.0.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/2.0;
        interface fe-1/2/1.0;
        interface lo0.0 {
            passive;
```

```
        }
    }
}
```

```
user@ISP-2# show policy-options
policy-statement exchange-peer {
    term AS64510-Aggregate {
        from {
            protocol aggregate;
            route-filter 172.16.32.0/21 exact;
        }
        then accept;
    }
    term Customer-2-Aggregate {
        from {
            protocol aggregate;
            route-filter 172.16.44.0/23 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement internal-peers {
    term statics {
        from protocol static;
        then accept;
    }
    term next-hop-self {
        then {
            next-hop self;
        }
    }
}
policy-statement private-peer {
    term statics {
        from protocol static;
        then accept;
    }
    term isp-and-customer-routes {
```

```
        from {
            protocol bgp;
            route-filter 172.16.32.0/21 orlonger;
        }
        then accept;
    }
    term reject-all {
        then reject;
    }
}
```

```
user@ISP-2# show routing-options
static {
    route 172.16.34.0/24 reject;
    route 172.16.35.0/24 reject;
}
aggregate {
    route 172.16.44.0/23;
    route 172.16.32.0/21;
}
router-id 192.168.0.2;
autonomous-system 64510;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device ISP-3

IN THIS SECTION

- Procedure | 166

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

On Device ISP-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 Device ISP-3 that represent local customers, these routes are sent as well as all other internal routes announced to the local router by the other ISP routers.

If the upstream route from Device Exchange-1 (172.16.8.0/21) is present, Device ISP-3 generates a default route.

To configure Device ISP-3:

1. Configure the device interfaces.

```
[edit interfaces]
user@ISP-3# set fe-1/2/0 unit 0 description to_ISP-1
user@ISP-3# set fe-1/2/0 unit 0 family inet address 10.0.0.1/30
user@ISP-3# set fe-1/2/2 unit 0 description to_ISP-2
user@ISP-3# set fe-1/2/2 unit 0 family inet address 10.0.0.5/30
user@ISP-3# set fe-1/2/3 unit 0 description to_Customer-1
user@ISP-3# set fe-1/2/3 unit 0 family inet address 10.1.0.5/30
user@ISP-3# set fe-1/2/1 unit 0 description to_Customer-2
user@ISP-3# set fe-1/2/1 unit 0 family inet address 10.0.0.9/30
user@ISP-3# set lo0 unit 0 family inet address 192.168.0.3/32
```

2. Configure the interior gateway protocol (IGP).

```
[edit protocols ospf area 0.0.0.0]
user@ISP-3# set interface fe-1/2/0.0
user@ISP-3# set interface fe-1/2/2.0
user@ISP-3# set interface lo0.0 passive
```

3. Configure the static routes.

```
[edit routing-options static]
user@ISP-3# set route 172.16.36.0/24 reject
user@ISP-3# set route 172.16.37.0/24 reject
user@ISP-3# set route 172.16.38.0/24 reject
user@ISP-3# set route 172.16.39.0/24 reject
```

4. Configure a routing policy that generates a default static route only if a certain upstream route exists.

```
[edit policy-options policy-statement if-upstream-routes-exist term only-certain-
contributing-routes]
user@ISP-3# set from route-filter 172.16.8.0/21 exact
user@ISP-3# set then accept
[edit policy-options policy-statement if-upstream-routes-exist]
user@ISP-3# set term reject-all-other-routes then reject
[edit routing-options generate route 0.0.0.0/0]
user@ISP-3# set policy if-upstream-routes-exist
```

5. Configure the routing policy for Customer-1.

```
[edit policy-options policy-statement customer-1-peer]
user@ISP-3# set term defaut-route from route-filter 0.0.0.0/0 exact
user@ISP-3# set term defaut-route then accept
user@ISP-3# set term reject-all-other-routes then reject
```

6. Configure the routing policy for Customer-2.

```
[edit policy-options policy-statement customer-2-peer]
user@ISP-3# set term statics from protocol static
user@ISP-3# set term statics then accept
user@ISP-3# set term isp-and-customer-routes from protocol bgp
user@ISP-3# set term isp-and-customer-routes from route-filter 172.16.32.0/21 orlonger
user@ISP-3# set term isp-and-customer-routes then accept
user@ISP-3# set term default-route from route-filter 0.0.0.0/0 exact
user@ISP-3# set term default-route then accept
user@ISP-3# set term reject-all-other-routes then reject
```

7. Configure the routing policies for the internal peers.

```
[edit policy-options policy-statement internal-peers]
user@ISP-3# set term statics from protocol static
user@ISP-3# set term statics then accept
user@ISP-3# set term next then next-hop self
```

8. Configure the internal BGP (IBGP) connections to the other ISP devices.

```
[edit protocols bgp group int]
user@ISP-3# set type internal
user@ISP-3# set local-address 192.168.0.3
user@ISP-3# set export internal-peers
user@ISP-3# set neighbor 192.168.0.1
user@ISP-3# set neighbor 192.168.0.2
```

9. Configure the EBGP connections to the customer peers.

```
[edit protocols bgp group to_64511]
user@ISP-3# set type external
user@ISP-3# set export customer-1-peer
user@ISP-3# set neighbor 10.1.0.6 peer-as 64511
[edit protocols bgp group to_64512]
user@ISP-3# set type external
user@ISP-3# set export customer-2-peer
user@ISP-3# set neighbor 10.0.0.10 peer-as 64512
```

10. Configure the autonomous system (AS) number and the router ID.

```
[edit routing-options]
user@ISP-3# set router-id 192.168.0.3
user@ISP-3# set autonomous-system 64510
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@ISP-3# show interfaces
fe-1/2/0 {
    unit 0 {
        description to_ISP-1;
        family inet {
            address 10.0.0.1/30;
```

```
        }
    }
}
fe-1/2/1 {
    unit 0 {
        description to_Customer-2;
        family inet {
            address 10.0.0.9/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        description to_ISP-2;
        family inet {
            address 10.0.0.5/30;
        }
    }
}
fe-1/2/3 {
    unit 0 {
        description to_Customer-1;
        family inet {
            address 10.1.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.3/32;
        }
    }
}
```

```
user@ISP-3# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.3;
        export internal-peers;
```

```
        neighbor 192.168.0.1;
        neighbor 192.168.0.2;
    }
    group to_64511 {
        type external;
        export customer-1-peer;
        neighbor 10.1.0.6 {
            peer-as 64511;
        }
    }
    group to_64512 {
        type external;
        export customer-2-peer;
        neighbor 10.0.0.10 {
            peer-as 64512;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/0.0;
        interface fe-1/2/2.0;
        interface lo0.0 {
            passive;
        }
    }
}
```

```
user@ISP-3# show policy-options
policy-statement customer-1-peer {
    term defaut-route {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement customer-2-peer {
```

```
    term statics {
        from protocol static;
        then accept;
    }
    term isp-and-customer-routes {
        from {
            protocol bgp;
            route-filter 172.16.32.0/21 orlonger;
        }
        then accept;
    }
    term default-route {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement if-upstream-routes-exist {
    term only-certain-contributing-routes {
        from {
            route-filter 172.16.8.0/21 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement internal-peers {
    term statics {
        from protocol static;
        then accept;
    }
    term next {
        then {
            next-hop self;
        }
```

```
    }
}
```

```
user@ISP-3# show routing-options
static {
    route 172.16.36.0/24 reject;
    route 172.16.37.0/24 reject;
    route 172.16.38.0/24 reject;
    route 172.16.39.0/24 reject;
}
generate {
    route 0.0.0.0/0 policy if-upstream-routes-exist;
}
router-id 192.168.0.3;
autonomous-system 64510;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device Exchange-2

IN THIS SECTION

- Procedure | 173

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

Device Exchange-2 exchanges all BGP routes with all BGP peers. The outbound-routes policy for Device Exchange-2 advertises locally defined static routes using BGP. The exclusion of a final `then reject` term causes the default BGP export policy to take effect, which is to send all BGP routes to all external BGP peers.

To configure Device Exchange-2:

1. Configure the device interfaces.

```
[edit interfaces]
user@Exchange-2# set fe-1/2/0 unit 0 description to_ISP-2
user@Exchange-2# set fe-1/2/0 unit 0 family inet address 10.3.0.1/30
user@Exchange-2# set fe-1/2/2 unit 0 description to_Exchange-1
user@Exchange-2# set fe-1/2/2 unit 0 family inet address 10.3.0.41/30
user@Exchange-2# set fe-1/2/1 unit 0 description to_Private-Peer-2
user@Exchange-2# set fe-1/2/1 unit 0 family inet address 10.3.0.49/30
user@Exchange-2# set lo0 unit 0 family inet address 192.168.0.7/32
```

2. Configure the static routes.

```
[edit routing-options static]
set route 172.16.16.0/21 reject
```

3. Configure a routing policy that generates a default static route only if certain internal routes exist.

```
[edit policy-options policy-statement outbound-routes term statics]
user@Exchange-2# set from protocol static
user@Exchange-2# set then accept
```

4. Configure the EBGP connections to the customer peers.

```
[edit protocols bgp group ext]
user@Exchange-2# set type external
user@Exchange-2# set export outbound-routes
user@Exchange-2# set neighbor 10.3.0.2 peer-as 64510
user@Exchange-2# set neighbor 10.3.0.50 peer-as 64516
user@Exchange-2# set neighbor 10.3.0.42 peer-as 64514
```

5. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@Exchange-2# set autonomous-system 64515
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@Exchange-2 show interfaces
fe-1/2/0 {
    unit 0 {
        description to_ISP-2;
        family inet {
            address 10.3.0.1/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        description to_Private-Peer-2;
        family inet {
            address 10.3.0.49/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        description to_Exchange-1;
        family inet {
            address 10.3.0.41/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.7/32;
        }
    }
}
```

```
user@Exchange-2# show protocols
bgp {
```

```
    group ext {
        type external;
        export outbound-routes;
        neighbor 10.3.0.2 {
            peer-as 64510;
        }
        neighbor 10.3.0.50 {
            peer-as 64516;
        }
        neighbor 10.3.0.42 {
            peer-as 64514;
        }
    }
}
```

```
user@Exchange-2# show policy-options
policy-statement outbound-routes {
    term statics {
        from protocol static;
        then accept;
    }
}
```

```
user@Exchange-2# show routing-options
static {
    route 172.16.16.0/21 reject;
}
autonomous-system 64515;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device Private-Peer-2

IN THIS SECTION

- Procedure | 177

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

Device Private-Peer-2 performs two main functions:

- Advertises routes local to AS 64516 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 64516 or AS 64512. These routes include other AS 64516 customer routes in addition to the AS 64512 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 64512 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 64512 router using BGP and the configured policies. The `if-upstream-routes-exist` policy accepts only the 172.16.32.0/21 route from Device Exchange-2, and rejects all other routes. If the 172.16.32.0/21 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 64512 customer router.

To configure Device Private-Peer-2:

1. Configure the device interfaces.

```
[edit interfaces]
user@Private-Peer-2# set fe-1/2/3 unit 0 description to_ISP-2
user@Private-Peer-2# set fe-1/2/3 unit 0 family inet address 10.3.0.5/30
user@Private-Peer-2# set fe-1/2/0 unit 0 description to_Customer-1
user@Private-Peer-2# set fe-1/2/0 unit 0 family inet address 10.0.0.22/30
user@Private-Peer-2# set fe-1/2/1 unit 0 description to_Exchange-2
user@Private-Peer-2# set fe-1/2/1 unit 0 family inet address 10.3.0.50/30
user@Private-Peer-2# set lo0 unit 0 family inet address 192.168.0.5/32
```

2. Configure the static routes.

```
[edit routing-options static]
user@Private-Peer-2# set route 172.16.24.0/25 reject
user@Private-Peer-2# set route 172.16.24.128/25 reject
```

```
user@Private-Peer-2# set route 172.16.25.0/26 reject
user@Private-Peer-2# set route 172.16.25.64/26 reject
```

3. Configure a routing policy that generates a default static route only if certain internal routes exist.

```
[edit policy-options policy-statement if-upstream-routes-exist]
user@Private-Peer-2# set term as-64515-routes from route-filter 172.16.16.0/21 exact
user@Private-Peer-2# set term as-64515-routes then accept
user@Private-Peer-2# set term reject-all-other-routes then reject
[edit routing-options generate route 0.0.0.0/0]
user@Private-Peer-2# set policy if-upstream-routes-exist
```

4. Configure the routing policy that advertises local static routes and the default route.

```
[edit policy-options policy-statement internal-routes]
user@Private-Peer-2# set term statics from protocol static
user@Private-Peer-2# set term statics then accept
user@Private-Peer-2# set term default-route from route-filter 0.0.0.0/0 exact
user@Private-Peer-2# set term default-route then accept
user@Private-Peer-2# set term reject-all-other-routes then reject
```

5. Configure the routing policy that advertises local customer routes.

```
[edit policy-options policy-statement outbound-routes]
user@Private-Peer-2# set term statics from protocol static
user@Private-Peer-2# set term statics then accept
user@Private-Peer-2# set term allowed-bgp-routes from as-path my-own-routes
user@Private-Peer-2# set term allowed-bgp-routes from as-path AS64512-routes
user@Private-Peer-2# set term allowed-bgp-routes then accept
user@Private-Peer-2# set term no-transit then reject
[edit policy-options]
user@Private-Peer-2# set as-path my-own-routes "()"
user@Private-Peer-2# set as-path AS64512-routes 64512
```

6. Configure the EBGP connection to Customer-2.

```
[edit protocols bgp group to-64512]
user@Private-Peer-2# set type external
user@Private-Peer-2# set export internal-routes
```

```
user@Private-Peer-2# set peer-as 64512
user@Private-Peer-2# set neighbor 10.0.0.21
```

7. Configure the EBGP connection to Device Exchange-2.

```
[edit protocols bgp group to-64515]
user@Private-Peer-2# set type external
user@Private-Peer-2# set export outbound-routes
user@Private-Peer-2# set peer-as 64515
user@Private-Peer-2# set neighbor 10.3.0.49
```

8. Configure the EBGP connections to the ISP.

```
[edit protocols bgp group ext]
user@Private-Peer-2# set type external
user@Private-Peer-2# set export outbound-routes
user@Private-Peer-2# set peer-as 64510
user@Private-Peer-2# set neighbor 10.3.0.6
```

9. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@Private-Peer-2# set autonomous-system 64516
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@Private-Peer-2# show interfaces
fe-1/2/0 {
    unit 0 {
        description to_Customer-1;
        family inet {
            address 10.0.0.22/30;
        }
    }
```

```
}
fe-1/2/1 {
    unit 0 {
        description to_Exchange-2;
        family inet {
            address 10.3.0.50/30;
        }
    }
}
fe-1/2/3 {
    unit 0 {
        description to_ISP-2;
        family inet {
            address 10.3.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.5/32;
        }
    }
}
```

```
user@Private-Peer-2# show protocols
bgp {
    group ext {
        type external;
        export outbound-routes;
        peer-as 64510;
        neighbor 10.3.0.6;
    }
    group to-64512 {
        type external;
        export internal-routes;
        peer-as 64512;
        neighbor 10.0.0.21;
    }
    group to-64515 {
        type external;
```

```
        export outbound-routes;
        peer-as 64515;
        neighbor 10.3.0.49;
    }
}
```

```
user@Private-Peer-2# show policy-options
policy-statement if-upstream-routes-exist {
    term as-64515-routes {
        from {
            route-filter 172.16.16.0/21 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement internal-routes {
    term statics {
        from protocol static;
        then accept;
    }
    term default-route {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term reject-all-other-routes {
        then reject;
    }
}
policy-statement outbound-routes {
    term statics {
        from protocol static;
        then accept;
    }
    term allowed-bgp-routes {
        from as-path [ my-own-routes AS64512-routes ];
        then accept;
```

```
    }
    term no-transit {
        then reject;
    }
}
as-path my-own-routes "()";
as-path AS64512-routes 64512;
```

```
user@Private-Peer-2# show routing-options
static {
    route 172.16.24.0/25 reject;
    route 172.16.24.128/25 reject;
    route 172.16.25.0/26 reject;
    route 172.16.25.64/26 reject;
}
generate {
    route 0.0.0.0/0 policy if-upstream-routes-exist;
}
autonomous-system 64516;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes on Device Customer-1 | **183**
- Verifying the Routes on Device Customer-2 | **184**
- Verifying the Routes on Device ISP-1 | **185**
- Verifying the Routes on Device ISP-2 | **189**
- Verifying the Routes on Device ISP-3 | **193**
- Verifying the Routes on Device Exchange-1 | **196**
- Verifying the Routes on Device Exchange-2 | **198**
- Verifying the Routes on Device Private-Peer-1 | **200**
- Verifying the Routes on Device Private-Peer-2 | **201**

Confirm that the configuration is working properly.

Verifying the Routes on Device Customer-1

Purpose

On Device Customer-1, check the routes in the routing table.

Action

```
user@Customer-1> show route

inet.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:09:25, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.1.0.5 via fe-1/2/3.0
10.1.0.4/30        *[Direct/0] 23:50:20
                    > via fe-1/2/3.0
10.1.0.6/32        *[Local/0] 5d 21:56:47
                      Local via fe-1/2/3.0
172.16.40.0/25     *[Static/5] 22:59:04
                      Reject
172.16.40.128/25   *[Static/5] 22:59:04
                      Reject
172.16.41.0/25     *[Static/5] 22:59:04
                      Reject
172.16.41.128/25   *[Static/5] 22:59:04
                      Reject
192.168.0.8/32     *[Direct/0] 5d 21:25:45
                    > via lo0.0
```

Meaning

Device Customer-1 has its four static routes, and it has learned the default route through BGP.

Verifying the Routes on Device Customer-2

Purpose

On Device Customer-2, check the routes in the routing table.

Action

```
user@Customer-2> show route
inet.0: 22 destinations, 23 routes (22 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:10:35, localpref 200
                      AS path: 64510 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/0.10
                    [BGP/170] 04:58:09, localpref 50
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.22 via fe-1/2/0.0
10.0.0.8/30        *[Direct/0] 23:51:29
                    > via fe-1/2/0.10
10.0.0.10/32       *[Local/0] 23:52:49
                      Local via fe-1/2/0.10
10.0.0.20/30       *[Direct/0] 23:52:49
                    > via fe-1/2/0.0
10.0.0.21/32       *[Local/0] 23:52:49
                      Local via fe-1/2/0.0
172.16.24.0/25     *[BGP/170] 04:58:09, localpref 50
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.22 via fe-1/2/0.0
172.16.24.128/25   *[BGP/170] 04:58:09, localpref 50
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.22 via fe-1/2/0.0
172.16.25.0/26     *[BGP/170] 04:58:09, localpref 50
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.22 via fe-1/2/0.0
172.16.25.64/26    *[BGP/170] 04:58:09, localpref 50
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.22 via fe-1/2/0.0
172.16.32.0/24     *[BGP/170] 22:38:47, localpref 200
                      AS path: 64510 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/0.10
172.16.33.0/24     *[BGP/170] 22:38:47, localpref 200
```

```
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.34.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.35.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.36.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.37.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.38.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.39.0/24     *[BGP/170] 22:38:47, localpref 200
                    AS path: 64510 I, validation-state: unverified
                  > to 10.0.0.9 via fe-1/2/0.10
172.16.44.0/26     *[Static/5] 22:57:28
                    Reject
172.16.44.64/26    *[Static/5] 22:57:28
                    Reject
172.16.44.128/26   *[Static/5] 22:57:28
                    Reject
172.16.44.192/26   *[Static/5] 22:57:28
                    Reject
192.168.0.9/32     *[Direct/0] 23:52:49
                  > via lo0.0
```

Meaning

Device Customer-2 has learned the default route through its session with the ISP and also through its session with the private peer. The route learned from the ISP is preferred because it has a higher local preference.

Verifying the Routes on Device ISP-1

Purpose

On Device ISP-1, check the routes in the routing table.

Action

```
user@ISP-1> show route
inet.0: 42 destinations, 53 routes (42 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 22:44:26, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
10.0.0.0/30        *[Direct/0] 23:52:01
                    > via fe-1/2/0.0
10.0.0.2/32        *[Local/0] 23:52:01
                      Local via fe-1/2/0.0
10.0.0.4/30        *[OSPF/10] 23:51:06, metric 2
                      to 10.1.0.1 via fe-1/2/1.0
                    > to 10.0.0.1 via fe-1/2/0.0
10.0.0.20/30       *[BGP/170] 23:50:55, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                    [BGP/170] 23:51:28, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
10.1.0.0/30        *[Direct/0] 23:52:01
                    > via fe-1/2/1.0
10.1.0.2/32        *[Local/0] 23:52:01
                      Local via fe-1/2/1.0
10.2.0.0/30        *[Direct/0] 23:52:01
                    > via fe-1/2/2.0
10.2.0.2/32        *[Local/0] 23:52:01
                      Local via fe-1/2/2.0
10.2.0.4/30        *[Direct/0] 23:52:00
                    > via fe-1/2/3.0
10.2.0.6/32        *[Local/0] 23:52:00
                      Local via fe-1/2/3.0
10.3.0.4/30        *[BGP/170] 23:51:28, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
10.3.0.48/30       *[BGP/170] 23:50:55, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
172.16.8.0/21      *[BGP/170] 00:11:08, localpref 100
                      AS path: 64514 I, validation-state: unverified
```

```
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.16.0/21      *[BGP/170] 02:02:10, localpref 100, from 192.168.0.2
                      AS path: 64515 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                     [BGP/170] 02:02:10, localpref 100
                      AS path: 64514 64515 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.24.0/25      *[BGP/170] 23:06:33, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                     [BGP/170] 23:06:33, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.24.128/25    *[BGP/170] 23:06:33, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                     [BGP/170] 23:06:33, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.25.0/26      *[BGP/170] 23:06:33, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                     [BGP/170] 23:06:33, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.25.64/26     *[BGP/170] 23:06:33, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                     [BGP/170] 23:06:33, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.32.0/21      *[Aggregate/130] 22:44:27
                      Reject
172.16.32.0/24      *[Static/5] 22:44:27
                      Reject
172.16.33.0/24      *[Static/5] 22:44:27
                      Reject
172.16.34.0/24      *[BGP/170] 22:39:20, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
172.16.35.0/24      *[BGP/170] 22:39:20, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
```

```
172.16.36.0/24     *[BGP/170] 22:39:20, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.37.0/24     *[BGP/170] 22:39:20, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.38.0/24     *[BGP/170] 22:39:20, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.39.0/24     *[BGP/170] 22:39:20, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.40.0/22     *[Aggregate/130] 22:44:27
                      Reject
172.16.40.0/25     *[BGP/170] 23:00:47, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.40.128/25   *[BGP/170] 23:00:47, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.41.0/25     *[BGP/170] 23:00:47, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.41.128/25   *[BGP/170] 23:00:47, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.44.0/26     *[BGP/170] 22:58:01, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
                    [BGP/170] 22:58:01, localpref 100
                      AS path: 64514 64515 64516 64512 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.44.64/26    *[BGP/170] 22:58:01, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
                    [BGP/170] 22:58:01, localpref 100
                      AS path: 64514 64515 64516 64512 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.44.128/26   *[BGP/170] 22:58:01, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
                    [BGP/170] 22:58:01, localpref 100
                      AS path: 64514 64515 64516 64512 I, validation-state: unverified
```

```
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.44.192/26   *[BGP/170] 22:58:01, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
                    [BGP/170] 22:58:01, localpref 100
                      AS path: 64514 64515 64516 64512 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
192.168.0.1/32     *[Direct/0] 23:52:01
                    > via lo0.0
192.168.0.2/32     *[OSPF/10] 23:51:06, metric 1
                    > to 10.1.0.1 via fe-1/2/1.0
192.168.0.3/32     *[OSPF/10] 23:51:06, metric 1
                    > to 10.0.0.1 via fe-1/2/0.0
192.168.0.5/32     *[BGP/170] 23:50:55, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.1.0.1 via fe-1/2/1.0
                    [BGP/170] 23:51:28, localpref 100
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.2.0.5 via fe-1/2/3.0
172.16.233.5/32      *[OSPF/10] 23:52:07, metric 1
                      MultiRecv
```

Verifying the Routes on Device ISP-2

Purpose

On Device ISP-2, check the routes in the routing table.

Action

```
user@ISP-2> show route
inet.0: 41 destinations, 59 routes (41 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 22:45:44, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
10.0.0.0/30        *[OSPF/10] 23:52:25, metric 2
                      to 10.0.0.5 via fe-1/2/2.0
                    > to 10.1.0.2 via fe-1/2/1.0
10.0.0.4/30        *[Direct/0] 23:53:21
```

```
                    > via fe-1/2/2.0
10.0.0.6/32         *[Local/0] 23:53:23
                      Local via fe-1/2/2.0
10.0.0.20/30        *[BGP/170] 23:53:11, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
                    [BGP/170] 23:53:09, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
10.1.0.0/30         *[Direct/0] 23:53:19
                    > via fe-1/2/1.0
10.1.0.1/32         *[Local/0] 23:53:23
                      Local via fe-1/2/1.0
10.3.0.0/30         *[Direct/0] 23:53:22
                    > via fe-1/2/0.0
10.3.0.2/32         *[Local/0] 23:53:23
                      Local via fe-1/2/0.0
10.3.0.4/30         *[Direct/0] 23:53:23
                    > via fe-1/2/3.0
                    [BGP/170] 23:53:11, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
                    [BGP/170] 23:53:09, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
                    [BGP/170] 23:52:13, localpref 100, from 192.168.0.1
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.1.0.2 via fe-1/2/1.0
10.3.0.6/32         *[Local/0] 23:53:23
                      Local via fe-1/2/3.0
10.3.0.48/30        *[BGP/170] 23:53:11, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
172.16.8.0/21       *[BGP/170] 00:12:26, localpref 100, from 192.168.0.1
                      AS path: 64514 I, validation-state: unverified
                    > to 10.1.0.2 via fe-1/2/1.0
                    [BGP/170] 00:12:26, localpref 100
                      AS path: 64515 64514 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
172.16.16.0/21      *[BGP/170] 02:03:28, localpref 100
                      AS path: 64515 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
172.16.24.0/25      *[BGP/170] 23:07:51, localpref 100
```

```
                    AS path: 64516 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 23:07:51, localpref 100
                    AS path: 64515 64516 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.24.128/25  *[BGP/170] 23:07:51, localpref 100
                    AS path: 64516 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 23:07:51, localpref 100
                    AS path: 64515 64516 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.25.0/26    *[BGP/170] 23:07:51, localpref 100
                    AS path: 64516 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 23:07:51, localpref 100
                    AS path: 64515 64516 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.25.64/26   *[BGP/170] 23:07:51, localpref 100
                    AS path: 64516 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 23:07:51, localpref 100
                    AS path: 64515 64516 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.32.0/21    *[Aggregate/130] 22:40:38
                    Reject
172.16.32.0/24    *[BGP/170] 22:45:44, localpref 100, from 192.168.0.1
                    AS path: I, validation-state: unverified
                  > to 10.1.0.2 via fe-1/2/1.0
172.16.33.0/24    *[BGP/170] 22:45:44, localpref 100, from 192.168.0.1
                    AS path: I, validation-state: unverified
                  > to 10.1.0.2 via fe-1/2/1.0
172.16.34.0/24    *[Static/5] 22:40:38
                    Reject
172.16.35.0/24    *[Static/5] 22:40:38
                    Reject
172.16.36.0/24    *[BGP/170] 22:40:38, localpref 100, from 192.168.0.3
                    AS path: I, validation-state: unverified
                  > to 10.0.0.5 via fe-1/2/2.0
172.16.37.0/24    *[BGP/170] 22:40:38, localpref 100, from 192.168.0.3
                    AS path: I, validation-state: unverified
                  > to 10.0.0.5 via fe-1/2/2.0
172.16.38.0/24    *[BGP/170] 22:40:38, localpref 100, from 192.168.0.3
                    AS path: I, validation-state: unverified
```

```
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.39.0/24      *[BGP/170] 22:40:38, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.40.0/25      *[BGP/170] 23:02:05, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.40.128/25    *[BGP/170] 23:02:05, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.41.0/25      *[BGP/170] 23:02:05, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.41.128/25    *[BGP/170] 23:02:05, localpref 100, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
172.16.44.0/23      *[Aggregate/130] 22:40:38
                      Reject
172.16.44.0/26      *[BGP/170] 22:59:19, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
                    [BGP/170] 22:59:19, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
                    [BGP/170] 22:59:19, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
172.16.44.64/26     *[BGP/170] 22:59:19, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
                    [BGP/170] 22:59:19, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
                    [BGP/170] 22:59:19, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
                    > to 10.3.0.1 via fe-1/2/0.0
172.16.44.128/26    *[BGP/170] 22:59:19, localpref 100, from 192.168.0.3
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.5 via fe-1/2/2.0
                    [BGP/170] 22:59:19, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.5 via fe-1/2/3.0
                    [BGP/170] 22:59:19, localpref 100
```

```
                    AS path: 64515 64516 64512 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.44.192/26  *[BGP/170] 22:59:19, localpref 100, from 192.168.0.3
                    AS path: 64512 I, validation-state: unverified
                  > to 10.0.0.5 via fe-1/2/2.0
                  [BGP/170] 22:59:19, localpref 100
                    AS path: 64516 64512 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 22:59:19, localpref 100
                    AS path: 64515 64516 64512 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
192.168.0.1/32    *[OSPF/10] 23:52:25, metric 1
                  > to 10.1.0.2 via fe-1/2/1.0
192.168.0.2/32    *[Direct/0] 23:53:23
                  > via lo0.0
192.168.0.3/32    *[OSPF/10] 23:52:30, metric 1
                  > to 10.0.0.5 via fe-1/2/2.0
192.168.0.5/32    *[BGP/170] 23:53:11, localpref 100
                    AS path: 64516 I, validation-state: unverified
                  > to 10.3.0.5 via fe-1/2/3.0
                  [BGP/170] 23:53:09, localpref 100
                    AS path: 64515 64516 I, validation-state: unverified
                  > to 10.3.0.1 via fe-1/2/0.0
172.16.233.5/32     *[OSPF/10] 23:53:25, metric 1
                    MultiRecv
```

Verifying the Routes on Device ISP-3

Purpose

On Device ISP-3, check the routes in the routing table.

Action

```
user@ISP-3> show route

inet.0: 40 destinations, 41 routes (40 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Aggregate/130] 23:53:57, metric2 1
                  > to 10.0.0.2 via fe-1/2/0.0
```

```
                    [BGP/170] 22:46:17, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
10.0.0.0/30         *[Direct/0] 23:53:52
                    > via fe-1/2/0.0
10.0.0.1/32         *[Local/0] 23:53:53
                      Local via fe-1/2/0.0
10.0.0.4/30         *[Direct/0] 23:53:54
                    > via fe-1/2/2.0
10.0.0.5/32         *[Local/0] 23:53:54
                      Local via fe-1/2/2.0
10.0.0.8/30         *[Direct/0] 23:53:53
                    > via fe-1/2/1.0
10.0.0.9/32         *[Local/0] 23:53:53
                      Local via fe-1/2/1.0
10.0.0.20/30        *[BGP/170] 23:53:02, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
10.1.0.0/30         *[OSPF/10] 23:53:03, metric 2
                    > to 10.0.0.6 via fe-1/2/2.0
                      to 10.0.0.2 via fe-1/2/0.0
10.1.0.4/30         *[Direct/0] 23:53:54
                    > via fe-1/2/3.0
10.1.0.5/32         *[Local/0] 23:53:54
                      Local via fe-1/2/3.0
10.3.0.4/30         *[BGP/170] 23:52:46, localpref 100, from 192.168.0.1
                      AS path: 64514 64515 64516 I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/2/0.0
10.3.0.48/30        *[BGP/170] 23:53:02, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.8.0/21       *[BGP/170] 00:12:59, localpref 100, from 192.168.0.1
                      AS path: 64514 I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/2/0.0
172.16.16.0/21      *[BGP/170] 02:04:01, localpref 100, from 192.168.0.2
                      AS path: 64515 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.24.0/25      *[BGP/170] 23:08:24, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.24.128/25    *[BGP/170] 23:08:24, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
```

```
172.16.25.0/26     *[BGP/170] 23:08:24, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.25.64/26    *[BGP/170] 23:08:24, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.32.0/24     *[BGP/170] 22:46:17, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/2/0.0
172.16.33.0/24     *[BGP/170] 22:46:17, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/2/0.0
172.16.34.0/24     *[BGP/170] 22:41:11, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.35.0/24     *[BGP/170] 22:41:11, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.36.0/24     *[Static/5] 22:41:11
                      Reject
172.16.37.0/24     *[Static/5] 22:41:11
                      Reject
172.16.38.0/24     *[Static/5] 22:41:11
                      Reject
172.16.39.0/24     *[Static/5] 22:41:11
                      Reject
172.16.40.0/25     *[BGP/170] 23:02:38, localpref 100
                      AS path: 64511 I, validation-state: unverified
                    > to 10.1.0.6 via fe-1/2/3.0
172.16.40.128/25   *[BGP/170] 23:02:38, localpref 100
                      AS path: 64511 I, validation-state: unverified
                    > to 10.1.0.6 via fe-1/2/3.0
172.16.41.0/25     *[BGP/170] 23:02:38, localpref 100
                      AS path: 64511 I, validation-state: unverified
                    > to 10.1.0.6 via fe-1/2/3.0
172.16.41.128/25   *[BGP/170] 23:02:38, localpref 100
                      AS path: 64511 I, validation-state: unverified
                    > to 10.1.0.6 via fe-1/2/3.0
172.16.44.0/26     *[BGP/170] 22:59:52, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.10 via fe-1/2/1.0
172.16.44.64/26    *[BGP/170] 22:59:52, localpref 100
                      AS path: 64512 I, validation-state: unverified
```

```
                    > to 10.0.0.10 via fe-1/2/1.0
172.16.44.128/26    *[BGP/170] 22:59:52, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.10 via fe-1/2/1.0
172.16.44.192/26    *[BGP/170] 22:59:52, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.10 via fe-1/2/1.0
192.168.0.1/32      *[OSPF/10] 23:53:03, metric 1
                    > to 10.0.0.2 via fe-1/2/0.0
192.168.0.2/32      *[OSPF/10] 23:53:03, metric 1
                    > to 10.0.0.6 via fe-1/2/2.0
192.168.0.3/32      *[Direct/0] 23:53:54
                    > via lo0.0
192.168.0.5/32      *[BGP/170] 23:53:02, localpref 100, from 192.168.0.2
                      AS path: 64516 I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
172.16.233.5/32       *[OSPF/10] 23:53:58, metric 1
                      MultiRecv
```

Verifying the Routes on Device Exchange-1

Purpose

On Device Exchange-1, check the routes in the routing table.

Action

```
user@Exchange-1> show route

inet.0: 23 destinations, 24 routes (23 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.20/30        *[BGP/170] 23:53:51, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
10.2.0.4/30         *[Direct/0] 23:54:23
                    > via fe-1/2/3.0
10.2.0.5/32         *[Local/0] 23:54:29
                      Local via fe-1/2/3.0
10.3.0.4/30         *[BGP/170] 23:53:51, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
```

```
                    > to 10.3.0.41 via fe-1/2/2.0
10.3.0.40/30        *[Direct/0] 23:54:27
                    > via fe-1/2/2.0
10.3.0.42/32        *[Local/0] 23:54:29
                      Local via fe-1/2/2.0
10.3.0.44/30        *[Direct/0] 23:54:29
                    > via fe-1/2/1.0
10.3.0.45/32        *[Local/0] 23:54:29
                      Local via fe-1/2/1.0
172.16.8.0/21       *[Static/5] 00:13:31
                      Reject
172.16.16.0/21      *[BGP/170] 02:04:33, localpref 100
                      AS path: 64515 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.24.0/25      *[BGP/170] 23:08:56, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.24.128/25    *[BGP/170] 23:08:56, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.25.0/26      *[BGP/170] 23:08:56, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.25.64/26     *[BGP/170] 23:08:56, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.32.0/21      *[BGP/170] 22:46:49, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.6 via fe-1/2/3.0
                    [BGP/170] 22:41:43, localpref 100
                      AS path: 64515 64510 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.40.0/22      *[BGP/170] 22:46:49, localpref 100
                      AS path: 64510 64511 I, validation-state: unverified
                    > to 10.2.0.6 via fe-1/2/3.0
172.16.44.0/23      *[BGP/170] 22:41:43, localpref 100
                      AS path: 64515 64510 64512 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.44.0/26      *[BGP/170] 23:00:24, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.44.64/26     *[BGP/170] 23:00:24, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
```

```
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.44.128/26    *[BGP/170] 23:00:24, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
172.16.44.192/26    *[BGP/170] 23:00:24, localpref 100
                      AS path: 64515 64516 64512 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
192.168.0.5/32      *[BGP/170] 23:53:51, localpref 100
                      AS path: 64515 64516 I, validation-state: unverified
                    > to 10.3.0.41 via fe-1/2/2.0
192.168.0.6/32      *[Direct/0] 23:54:29
                    > via lo0.0
```

Verifying the Routes on Device Exchange-2

Purpose

On Device Exchange-2, check the routes in the routing table.

Action

```
user@Exchange-2> show route
inet.0: 24 destinations, 26 routes (23 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.20/30        *[BGP/170] 23:54:44, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
10.3.0.0/30         *[Direct/0] 23:54:57
                    > via fe-1/2/0.0
10.3.0.1/32         *[Local/0] 23:54:57
                      Local via fe-1/2/0.0
10.3.0.4/30         *[BGP/170] 23:54:44, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
10.3.0.40/30        *[Direct/0] 23:54:57
                    > via fe-1/2/2.0
10.3.0.41/32        *[Local/0] 23:54:57
                      Local via fe-1/2/2.0
10.3.0.48/30        *[Direct/0] 23:54:57
                    > via fe-1/2/1.0
```

```
                    [BGP/170] 23:54:44, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
10.3.0.49/32        *[Local/0] 23:54:57
                      Local via fe-1/2/1.0
172.16.8.0/21       *[BGP/170] 00:14:01, localpref 100
                      AS path: 64514 I, validation-state: unverified
                    > to 10.3.0.42 via fe-1/2/2.0
172.16.16.0/21      *[Static/5] 02:05:03
                      Reject
172.16.24.0/25      *[BGP/170] 23:09:26, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.24.128/25    *[BGP/170] 23:09:26, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.25.0/26      *[BGP/170] 23:09:26, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.25.64/26     *[BGP/170] 23:09:26, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.32.0/21      *[BGP/170] 22:42:13, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.3.0.2 via fe-1/2/0.0
                    [BGP/170] 22:47:19, localpref 100
                      AS path: 64514 64510 I, validation-state: unverified
                    > to 10.3.0.42 via fe-1/2/2.0
172.16.40.0/22      *[BGP/170] 22:47:19, localpref 100
                      AS path: 64514 64510 64511 I, validation-state: unverified
                    > to 10.3.0.42 via fe-1/2/2.0
172.16.44.0/23      *[BGP/170] 22:42:13, localpref 100
                      AS path: 64510 64512 I, validation-state: unverified
                    > to 10.3.0.2 via fe-1/2/0.0
172.16.44.0/26      *[BGP/170] 23:00:54, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.44.64/26     *[BGP/170] 23:00:54, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
172.16.44.128/26    *[BGP/170] 23:00:54, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
```

```
172.16.44.192/26   *[BGP/170] 23:00:54, localpref 100
                      AS path: 64516 64512 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
192.168.0.5/32     *[BGP/170] 23:54:44, localpref 100
                      AS path: 64516 I, validation-state: unverified
                    > to 10.3.0.50 via fe-1/2/1.0
192.168.0.7/32     *[Direct/0] 23:54:57
                    > via lo0.0
```

Meaning

On Device Exchange-2, the default route 0/0 is hidden because the next hop for the route is its own interface to Device Private-Peer-2, from which the route was received. The route is hidden to avoid a loop.

Verifying the Routes on Device Private-Peer-1

Purpose

On Device Private-Peer-1, check the routes in the routing table.

Action

```
user@Private-Peer-1> show route

inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.0.0/30        *[Direct/0] 23:58:57
                    > via fe-1/2/2.0
10.2.0.1/32        *[Local/0] 5d 21:34:22
                      Local via fe-1/2/2.0
10.3.0.44/30       *[Direct/0] 23:59:02
                    > via fe-1/2/1.0
10.3.0.46/32       *[Local/0] 1d 03:19:52
                      Local via fe-1/2/1.0
172.16.32.0/24     *[BGP/170] 22:51:22, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.33.0/24     *[BGP/170] 22:51:22, localpref 100
                      AS path: 64510 I, validation-state: unverified
```

```
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.34.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.35.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.36.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.37.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.38.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
172.16.39.0/24      *[BGP/170] 22:46:16, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.2.0.2 via fe-1/2/2.0
192.168.0.4/32      *[Direct/0] 5d 21:34:22
                    > via lo0.0
```

Verifying the Routes on Device Private-Peer-2

Purpose

On Device Private-Peer-2, check the routes in the routing table.

Action

```
user@Private-Peer-2> show route


inet.0: 29 destinations, 29 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0           *[Aggregate/130] 1d 02:13:28
                    > to 10.3.0.49 via fe-1/2/1.0
10.0.0.20/30        *[Direct/0] 1d 00:00:53
                    > via fe-1/2/0.0
10.0.0.22/32        *[Local/0] 4d 23:51:14
```

```
                    Local via fe-1/2/0.0
10.3.0.4/30         *[Direct/0] 23:59:36
                     > via fe-1/2/3.0
10.3.0.5/32         *[Local/0] 5d 21:34:57
                    Local via fe-1/2/3.0
10.3.0.48/30        *[Direct/0] 23:59:35
                     > via fe-1/2/1.0
10.3.0.50/32        *[Local/0] 1d 03:20:27
                    Local via fe-1/2/1.0
172.16.8.0/21       *[BGP/170] 00:18:39, localpref 100
                    AS path: 64515 64514 I, validation-state: unverified
                     > to 10.3.0.49 via fe-1/2/1.0
172.16.16.0/21      *[BGP/170] 02:09:41, localpref 100
                    AS path: 64515 I, validation-state: unverified
                     > to 10.3.0.49 via fe-1/2/1.0
172.16.24.0/25      *[Static/5] 23:14:04
                    Reject
172.16.24.128/25    *[Static/5] 23:14:04
                    Reject
172.16.25.0/26      *[Static/5] 23:14:04
                    Reject
172.16.25.64/26     *[Static/5] 23:14:04
                    Reject
172.16.32.0/21      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64515 64510 I, validation-state: unverified
                     > to 10.3.0.49 via fe-1/2/1.0
172.16.32.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
                     > to 10.3.0.6 via fe-1/2/3.0
172.16.33.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
                     > to 10.3.0.6 via fe-1/2/3.0
172.16.34.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
                     > to 10.3.0.6 via fe-1/2/3.0
172.16.35.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
                     > to 10.3.0.6 via fe-1/2/3.0
172.16.36.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
                     > to 10.3.0.6 via fe-1/2/3.0
172.16.37.0/24      *[BGP/170] 22:46:51, localpref 100
                    AS path: 64510 I, validation-state: unverified
```

```
                    > to 10.3.0.6 via fe-1/2/3.0
172.16.38.0/24      *[BGP/170] 22:46:51, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.3.0.6 via fe-1/2/3.0
172.16.39.0/24      *[BGP/170] 22:46:51, localpref 100
                      AS path: 64510 I, validation-state: unverified
                    > to 10.3.0.6 via fe-1/2/3.0
172.16.40.0/22      *[BGP/170] 22:51:57, localpref 100
                      AS path: 64515 64514 64510 64511 I, validation-state: unverified
                    > to 10.3.0.49 via fe-1/2/1.0
172.16.44.0/23      *[BGP/170] 22:46:51, localpref 100
                      AS path: 64515 64510 64512 I, validation-state: unverified
                    > to 10.3.0.49 via fe-1/2/1.0
172.16.44.0/26      *[BGP/170] 23:05:32, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.21 via fe-1/2/0.0
172.16.44.64/26     *[BGP/170] 23:05:32, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.21 via fe-1/2/0.0
172.16.44.128/26    *[BGP/170] 23:05:32, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.21 via fe-1/2/0.0
172.16.44.192/26    *[BGP/170] 23:05:32, localpref 100
                      AS path: 64512 I, validation-state: unverified
                    > to 10.0.0.21 via fe-1/2/0.0
192.168.0.5/32      *[Direct/0] 5d 21:34:57
                    > via lo0.0
```

RELATED DOCUMENTATION

Example: Configuring Policy Chains and Route Filters | 258

Example: Configuring Routing Policy Prefix Lists | 396

Understanding Policy Expressions

IN THIS SECTION

- Policy Expression Examples | 206
- Policy Expression Evaluation | 207
- Evaluating Policy Expressions | 208

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 12 on page 204 summarizes the policy actions and their corresponding TRUE and FALSE values and flow control action values. Table 13 on page 205 describes the logical operators. For complete information about policy expression evaluation, see "Policy Expression Evaluation" on page 207.

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 12: 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 13: Policy Expression Logical Operators

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

For more information, see the following sections:

Policy Expression Examples

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 "Policy Expression Evaluation" on page 207.) 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]
```

Policy Expression Evaluation

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 12 on page 204.)

2. The software takes the value of TRUE or FALSE and evaluates it against the logical operator used in the policy expression (see Table 13 on page 205). 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 12 on page 204.) 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.

Evaluating Policy Expressions

The following sample routing policy uses three policy expressions:

```
[edit]
policy-options {
    policy-statement policy-A {
        from {
            route-filter 10.10.0.0/16 orlonger;
        }
        then reject;
    }
}
policy-options {
    policy-statement policy-B {
        from {
            route-filter 10.20.0.0/16 orlonger;
        }
        then accept;
    }
}
protocols {
    bgp {
        neighbor 192.168.1.1 {
            export (policy-A && policy-B);
        }
        neighbor 192.168.2.1 {
```

```
            export (policy-A || policy-B);
        }
        neighbor 192.168.3.1 {
            export (!policy-A);
        }
    }
}
```

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

- (policy-A && policy-B)—10.10.1.0/24 is evaluated against `policy-A`. 10.10.1.0/24 matches the route list specified in `policy-A`, so the specified action of `reject` is returned. `reject` is converted to a value of FALSE, and FALSE is evaluated against the specified logical AND. Because the result of FALSE is certain no matter what the results of the evaluation of `policy-B` are (in policy expression logic, any result AND a value of FALSE produces the output of FALSE), `policy-B` is not evaluated and the output of FALSE is produced. The FALSE output is converted to `reject`, and 10.10.1.0/24 is rejected.

- (policy-A || policy-B)—10.10.1.0/24 is evaluated against `policy-A`. 10.10.1.0/24 matches the route list specified in `policy-A`, so the specified action of `reject` is returned. `reject` is converted to a value of FALSE, then FALSE is evaluated against the specified logical OR. Because logical OR requires at least one value of TRUE to produce an output of TRUE, 10.10.1.0/24 is evaluated against `policy-B`. 10.10.1.0/24 does not match `policy-B`, so the default action of `next-policy` is returned. The `next-policy` is converted to a value of TRUE, then the value of FALSE (for `policy-A` evaluation) and TRUE (for `policy-B` evaluation) are evaluated against the specified logical OR. In policy expression logic, FALSE OR TRUE produce an output of TRUE. The output of TRUE is converted to `next-policy`. (TRUE is converted to `next-policy` because `next-policy` was the last action retained by the policy framework software.) `policy-B` is the last routing policy in the policy expression, so the action specified by the default export policy for BGP is taken.

- (!policy-A)—10.10.1.0/24 is evaluated against `policy-A`. 10.10.1.0/24 matches the route list specified in `policy-A`, so the specified action of `reject` is returned. `reject` is converted to a value of FALSE, and FALSE is evaluated against the specified logical NOT. The value of FALSE is reversed to an output of TRUE based on the rules of logical NOT. The output of TRUE is converted to `accept`, and route 10.10.1.0/24 is accepted.

RELATED DOCUMENTATION

Example: Testing a Routing Policy with Complex Regular Expressions | **760**

Example: Configuring a Policy Subroutine | **289**

Example: Configuring Policy Chains and Route Filters | **258**

Example: Configuring Routing Policy Prefix Lists | **396**

Understanding Backup Selection Policy for OSPF Protocol

Support for OSPF loop-free alternate (LFA) routes essentially adds IP fast-reroute capability for OSPF. Junos OS precomputes multiple loop-free backup routes for all OSPF routes. These backup routes are pre-installed in the Packet Forwarding Engine, which performs a local repair and implements the backup path when the link for a primary next hop for a particular route is no longer available. The selection of LFA is done randomly by selecting any matching LFA to progress to the given destination. This does not ensure best backup coverage available for the network. In order to choose the best LFA, Junos OS allows you to configure network-wide backup selection policies for each destination (IPv4 and IPv6) and a primary next-hop interface. These policies are evaluated based on admin-group, srlg, bandwidth, protection-type, metric, and node information.

During backup shortest-path-first (SPF) computation, each node and link attribute of the backup path is accumulated by IGP and is associated with every node (router) in the topology. The next hop in the best backup path is selected as the backup next hop in the routing table. In general, backup evaluation policy rules are categorized into the following types:

- Pruning — Rules configured to select the eligible backup path.
- Ordering — Rules configured to select the best among the eligible backup paths.

The backup selection policies can be configured with both pruning and ordering rules. While evaluating the backup policies, each backup path is assigned a score, an integer value that signifies the total weight of the evaluated criteria. The backup path with the highest score is selected.

To enforce LFA selection, configure various rules for the following attributes:

- admin-group– Administrative groups, also known as link coloring or resource class, are manually assigned attributes that describe the “color” of links, such that links with the same color conceptually belong to the same class. These configured administrative groups are defined under protocol MPLS. You can use administrative groups to implement a variety of backup selection policies using exclude, include-all, include-any, or preference.
- srlg— A shared risk link group (SRLG) is a set of links sharing a common resource, which affects all links in the set if the common resource fails. These links share the same risk of failure and are therefore considered to belong to the same SRLG. For example, links sharing a common fiber are said to be in the same SRLG because a fault with the fiber might cause all links in the group to fail. An SRLG is represented by a 32-bit number unique within an IGP (OSPF) domain. A link might belong to multiple SRLGs. You can define the backup selection to either allow or reject the common SRLGs between the primary and the backup path. This rejection of common SRLGs are based on the non-existence of link having common SRLGs in the primary next-hop and the backup SPF.

NOTE: Administrative groups and SRLGs can be created only for default topologies.

- bandwidth—The bandwidth specifies the bandwidth constraints between the primary and the backup path. The backup next-hop link can be used only if the bandwidth of the backup next-hop interface is greater than or equal to the bandwidth of the primary next hop.
- protection-type— The protection-type protects the destination from node failure of the primary node or link failure of the primary link. You can configure node, link, or node-link to protect the destination. If link-node is configured , then the node-protecting LFA is preferred over link-protection LFA.
- node- The node is per-node policy information. Here, node can be a directly connected router, remote router like RSVP backup LSP tail-end, or any other router in the backup SPF path. The nodes are identified through the route-id advertised by a node in the LSP. You can list the nodes to either prefer or exclude them in the backup path.
- metric— Metric decides how the LFAs should be preferred. In backup selection path, root metric and dest-metric are the two types of metrics. root-metric indicates the metric to the one-hop neighbor or a remote router such as an RSVP backup LSP tail-end router. The dest-metric indicates the metric from a one-hop neighbor or remote router such as an RSVP backup LSP tail-end router to the final destination. The metric evaluation is done either in ascending or descending order. By default, the first preference is given to backup paths with lowest destination evaluation and then to backup paths with lowest root metrics.

The evaluation-order allows you to control the order and criteria of evaluating these attributes in the backup path. You can explicitly configure the evaluation order. Only the configured attributes influence the backup path selection. The default order of evaluation of these attributes for the LFA is [admin-group srlg bandwidth protection-type node metric] .

NOTE: TE attributes are not supported in OSPFv3 and cannot be used for backup selection policy evaluation for IPv6 prefixes.

RELATED DOCUMENTATION

backup-selection (Protocols IS-IS)

Configuring Backup Selection Policy for the OSPF Protocol

Support for OSPF loop-free alternate (LFA) routes essentially adds IP fast-reroute capability for OSPF. Junos OS precomputes multiple loop-free backup routes for all OSPF routes. These backup routes are pre-installed in the Packet Forwarding Engine, which performs a local repair and implements the backup

path when the link for a primary next hop for a particular route is no longer available. The selection of LFA is done randomly by selecting any matching LFA to progress to the given destination. This does not ensure best backup coverage available for the network. In order to choose the best LFA, Junos OS allows you to configure network-wide backup selection policies for each destination (IPv4 and IPv6) and a primary next-hop interface. These policies are evaluated based on admin-group, srlg, bandwidth, protection-type, metric, and node information.

Before you begin to configure the backup selection policy for the OSPF protocol:

- Configure the router interfaces. See the *Junos OS Network Management Administration Guide for Routing Devices*.
- Configure an interior gateway protocol or static routing. See the *Junos OS Routing Protocols Library for Routing Devices*.

To configure the backup selection policy for the OSPF protocol:

1. Configure per-packet load balancing.

```
[edit policy-options]
 user@host# set policy-statement ecmp term 1 then load-balance per-packet
```

2. Enable RSVP on all the interfaces.

```
[edit protocols]
user@host# set rsvp interface all
```

3. Configure administrative groups.

```
[edit protocols mpls]
user@host# set admin-groups group-name
```

4. Configure srlg values.

```
[edit routing-options]
user@host# set srlg srlg-name srlg-value srlg-value
```

5. Enable MPLS on all the interfaces.

```
[edit protocols mpls]
user@host# set interface all
```

6. Apply MPLS to an interface configured with an administrative group.

```
[edit protocols mpls]
user@host# set interface interface-name admin-group group-name
```

7. Configure the ID of the router.

```
[edit routing-options]
user@host# set router-id router-id
```

8. Apply the routing policy to all equal cost multipaths exported from the routing table to the forwarding table.

```
[edit routing-options]
user@host# set forwarding-table export ecmp
```

9. Enable link protection and configure metric values on all the interfaces for an area.

```
[edit protocols ospf]
user@host# set area area-id interface interface-name link-protection
user@host# set area area-id interface interface-name metric metric
```

10. Configure the administrative group of the backup selection policy for an IP address.

    You can choose to exclude, include all, include any, or prefer the administrative groups from the backup path.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name admin-group
```

- Specify the administrative group to be excluded.

```
[edit routing-options backup-selection destination ip-address interface interface-name
admin-group]
user@host# set exclude group-name
```

  The backup path is not selected as the loop-free alternate (LFA) or backup nexthop if any of the links in the path have any one of the listed administrative groups.

For example, to exclude the group c1 from the administrative group:

```
[edit routing-options backup-selection destination 0.0.0.0/0 interface all admin-group]
user@host# set exclude c1
```

- Configure all the administrative groups if each link in the backup path requires all the listed administrative groups in order to accept the path.

```
[edit routing-options backup-selection destination ip-address interface interface-name
admin-group]
user@host# set include-all group-name
```

For example, to set all the administrative groups if each link requires all the listed administrative groups in order to accept the path:

```
[edit routing-options backup-selection destination 0.0.0.0/0 interface all admin-group]
user@host# set include-all c2
```

- Configure any administrative group if each link in the backup path requires at least one of the listed administrative groups in order to select the path.

```
[edit routing-options backup-selection destination ip-address interface interface-name
admin-group]
user@host# set include-any group-name
```

For example, to set any administrative group if each link in the backup path requires at least one of the listed administrative groups in order to select the path:

```
[edit routing-options backup-selection destination 0.0.0.0/0 interface all admin-group]
user@host# set include-any c3
```

- Define an ordered set of an administrative group that specifies the preference of the backup path.

The leftmost element in the set is given the highest preference.

```
[edit routing-options backup-selection destination ip-address interface interface-name
admin-group]
user@host#  set preference group-name
```

For example, to set an ordered set of an administrative group that specifies the preference of the backup path:

```
[edit routing-options backup-selection destination 0.0.0.0/0 interface all admin-group]
user@host#  set preference c4
```

11. Configure the backup path to allow the selection of the backup next hop only if the bandwidth is greater than or equal to the bandwidth of the primary next hop.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name bandwidth-
greater-equal-primary
```

12. Configure the backup path to specify the metric from the one-hop neighbor or from the remote router such as an RSVP backup label-switched-path (LSP) tail-end router to the final destination. The destination metric can be either highest or lowest.

    - Configure the backup path that has the highest destination metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name dest-
metric highest
```

    - Configure the backup path that has the lowest destination metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name dest-
metric lowest
```

13. Configure the backup path that is a downstream path to the destination.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name downstream-paths-only
```

14. Set the order of preference of the root and the destination metric during backup path selection.

    The preference order can be :

    - [root dest] — Backup path selection or preference is first based on the root-metric criteria. If the criteria of all the root-metric is the same, then the selection or preference is based on the dest-metric.
    - [dest root] — Backup path selection or preference is first based on the dest-metric criteria. If the criteria of all the dest-metric is the same, then the selection is based on the root-metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name metric-order dest
user@host# set backup-selection destination ip-address interface interface-name metric-order root
```

15. Configure the backup path to define a list of loop-back IP addresses of the adjacent neighbors to either exclude or prefer in the backup path selection.

    The neighbor can be a local (adjacent router) neighbor, remote neighbor, or any other router in the backup path.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name node
```

    - Configure the list of neighbors to be excluded.

```
[edit routing-options backup-selection destination ip-address interface interface-name node]
user@host# set exclude node-address
```

      The backup path that has a router from the list is not selected as the loop-free alternative or backup next hop.

- Configure an ordered set of neighbors to be preferred.

```
[edit routing-options backup-selection destination ip-address interface interface-name
node]
user@host# set preference node-address
```

The backup path having the leftmost neighbor is selected.

16. Configure the backup path to specify the required protection type of the backup path to be link, node, or node-link.

- Select the backup path that provides link protection.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name
protection-type link
```

- Select the backup path that provides node protection.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name
protection-type node
```

- Select the backup path that allows either node or link protection LFA where node-protection LFA is preferred over link-protection LFA.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface interface-name
protection-type node-link
```

17. Specify the metric to the one-hop neighbor or to the remote router such as an RSVP backup label-switched-path (LSP) tail-end router.

- Select the path with highest root metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface all root-metric highest
```

- Select the path with lowest root metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface all root-metric lowest
```

18. Configure the backup selection path to either allow or reject the common shared risk link groups (SRLGs) between the primary link and each link in the backup path.

    - Configure the backup path to allow common srlgs between the primary link and each link in the backup path.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface all srlg loose
```

      A backup path with a fewer number of srlg collisions is preferred.

    - Configure the backup path to reject the backup path that has common srlgs between the primary next-hop link and each link in the backup path.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface all srlg strict
```

19. Configure the backup path to control the order and the criteria of evaluating the backup path based on the administrative group, srlg, bandwidth, protection type, node, and metric.

    The default order of evaluation is admin-group, srlg, bandwidth, protection-type, node, and metric.

```
[edit routing-options]
user@host# set backup-selection destination ip-address interface all evaluation-order admin-
group
user@host# set backup-selection destination ip-address interface all evaluation-order srlg
user@host# set backup-selection destination ip-address interface all evaluation-order
bandwidth
```

RELATED DOCUMENTATION

backup-selection (Protocols IS-IS)

Configuring Backup Selection Policy for IS-IS Protocol

IN THIS SECTION

- Understanding Backup Selection Policy for IS-IS Protocol | 219

Understanding Backup Selection Policy for IS-IS Protocol

Support for IS-IS loop-free alternate (LFA) routes essentially adds IP fast-reroute capability for IS-IS. Junos OS precomputes multiple loop-free backup routes for all IS-IS routes. These backup routes are pre-installed in the Packet Forwarding Engine, which performs a local repair and implements the backup path when the link for a primary next hop for a particular route is no longer available. The selection of LFA is done randomly by selecting any matching LFA to progress to the given destination. This does not ensure best backup coverage available for the network. In order to choose the best LFA, Junos OS allows you to configure network-wide backup selection policies for each destination (IPv4 and IPv6) and a primary next-hop interface. These policies are evaluated based on admin-group, srlg, bandwidth, protection-type, metric, and neighbor information.

During backup shortest-path-first (SPF) computation, each node and link attribute of the backup path is accumulated by IGP and is associated with every node (router) in the topology. The next hop in the best backup path is selected as the backup next hop in the routing table. In general, backup evaluation policy rules are categorized into the following types:

- Pruning — Rules configured to select the eligible backup path.
- Ordering — Rules configured to select the best among the eligible backup paths.

The backup selection policies can be configured with both pruning and ordering rules. While evaluating the backup policies, each backup path is assigned a score, an integer value that signifies the total weight of the evaluated criteria. The backup path with the highest score is selected.

To enforce LFA selection, configure various rules for the following attributes:

- admin-group– Administrative groups, also known as link coloring or resource class, are manually assigned attributes that describe the “color” of links, such that links with the same color conceptually belong to the same class. These configured administrative groups are defined under protocol MPLS. You can use administrative groups to implement a variety of backup selection policies using exclude, include-all, include-any, or preference.
- backup-neighbor— A neighbor ID to either prefer or exclude in the backup path selection.

- node— A list of loop-back IP addresses of the adjacent nodes to either prefer or exclude in the backup path selection. The node can be a local (adjacent router) node, remote node, or any other router in the backup path. The nodes are identified through the TE-router-ID TLV advertised by a node in the LSP.
- node-tag— A node tag identifies a group of nodes in the network based on criteria such as the same neighbor tag values for all PE nodes to either prefer or exclude in the a backup path selection. This is implemented using IS-IS admin-tags. The routers are not identified with the explicit router-id but with an admin-tag prefix to their lo0 address prefix. These tags are advertised as part of extended IP reachability with a /32 prefix length that represents the TE-router _ID or node-ID of a router.
- srlg— A shared risk link group (SRLG) is a set of links sharing a common resource, which affects all links in the set if the common resource fails. These links share the same risk of failure and are therefore considered to belong to the same SRLG. For example, links sharing a common fiber are said to be in the same SRLG because a fault with the fiber might cause all links in the group to fail. An SRLG is represented by a 32-bit number unique within an IGP (IS-IS) domain. A link might belong to multiple SRLGs. You can define the backup selection to either allow or reject the common SRLGs between the primary and the backup path.
- bandwidth—The bandwidth specifies the bandwidth constraints between the primary and the backup path. The backup next-hop link can be used only if the bandwidth of the backup next-hop interface is greater than or equal to the bandwidth of the primary next hop.
- protection-type— The protection-type protects the destination from node failure of the primary node or link failure of the primary link. You can configure node, link, or node-link to protect the destination.. If link-node is configured , then the node-protecting LFA is preferred over link-protection LFA.
- metric— Metric decides how the LFAs should be preferred. In backup selection path, root metric and dest-metric are the two types of metrics. root-metric indicates the metric to the one-hop neighbor or a remote router such as an RSVP backup LSP tail-end router. The dest-metric indicates the metric from a one-hop neighbor or remote router such as an RSVP backup LSP tail-end router to the final destination. The metric evaluation is done either in ascending or descending order. By default, the first preference is given to backup paths with lowest destination evaluation and then to backup paths with lowest root metrics.

The evaluation-order allows you to control the order and criteria of evaluating these attributes in the backup path. You can explicitly configure the evaluation order. Only the configured attributes influence the backup path selection. The default order of evaluation of these attributes for the LFA is [admin-group srlg bandwidth protection-type neighbor neighbor-tag metric] .

SEE ALSO

> Example: Configuring Backup Selection Policy for IS-IS Protocol

backup-selection (Protocols IS-IS)

Example: Configuring Backup Selection Policy for the OSPF or OSPF3 Protocol

IN THIS SECTION

- Requirements | 221
- Overview | 222
- Configuration | 223
- Verification | 248

This example shows how to configure the backup selection policy for the OSPF or OSPF3 protocol, which enables you to select a loop-free alternate (LFA) in the network.

When you enable backup selection policies, Junos OS allows selection of LFA based on the policy rules and attributes of the links and nodes in the network. These attributes are admin-group, srlg, bandwidth, protection-type, metric, and node.

Requirements

This example uses the following hardware and software components:

- Eight routers that can be a combination of M Series Multiservice Edge Routers, MX Series 5G Universal Routing Platforms, PTX Series Packet Transport Routers, and T Series Core Routers
- Junos OS Release 15.1 or later running on all devices

Before you begin:

1. Configure the device interfaces.
2. Configure OSPF.

Overview

IN THIS SECTION

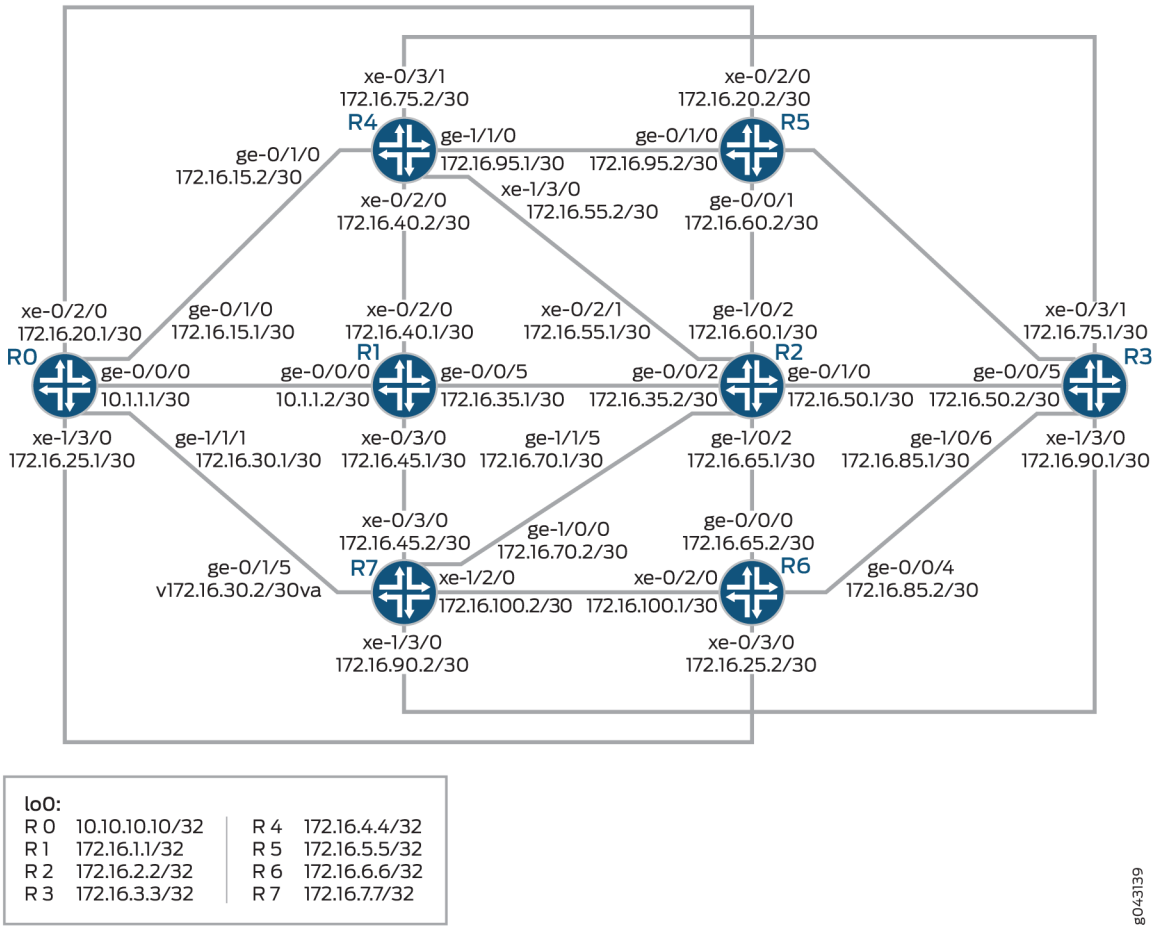
- Topology | 222

In Junos OS, the default loop-free alternative (LFA) selection algorithm or criteria can be overridden with an LFA policy. These policies are configured for each destination (IPv4 and IPv6) and a primary next-hop interface . These backup policies enforce LFA selection based on admin-group, srlg, bandwidth, protection-type, metric, and node attributes of the backup path. During backup shortest-path-first (SPF) computation, each attribute (both node and link) of the backup path, stored per backup next-hop, is accumulated by IGP. For the routes created internally by IGP, the attribute set of every backup path is evaluated against the policy configured for each destination (IPv4 and IPv6) and a primary next-hop interface. The first or the best backup path is selected and installed as the backup next hop in the routing table. To configure the backup selection policy, include the `backup-selection` configuration statement at the `[edit routing-options]` hierarchy level. The `show backup-selection` command displays the configured policies for a given interface and destination. The display can be filtered against a particular destination, prefix, interface, or logical systems.

Topology

In this topology shown in Figure 14 on page 223, the backup selection policy is configured on Device R3.

Figure 14: Example Backup Selection Policy for OSPF or OPSF3



Configuration

IN THIS SECTION

- CLI Quick Configuration | **224**
- Configuring Device R3 | **238**
- Results | **242**

CLI Quick Configuration

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

R0

```
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:10:1:1::1/64
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/1/0 unit 0 family inet address 172.16.15.1/30
set interfaces ge-0/1/0 unit 0 family inet6 address 2001:db8:15:1:1::1/64
set interfaces ge-0/1/0 unit 0 family mpls
set interfaces xe-0/2/0 unit 0 family inet address 172.16.20.1/30
set interfaces xe-0/2/0 unit 0 family inet6 address 2001:db8:20:1:1::1/64
set interfaces xe-0/2/0 unit 0 family mpls
set interfaces ge-1/0/5 unit 0 family inet address 172.16.150.1/24
set interfaces ge-1/0/5 unit 0 family inet6 address 2001:db8:150:1:1::1/64
set interfaces ge-1/0/5 unit 0 family mpls
set interfaces ge-1/1/1 unit 0 family inet address 172.16.30.1/30
set interfaces ge-1/1/1 unit 0 family inet6 address 2001:db8:30:1:1::1/64
set interfaces ge-1/1/1 unit 0 family mpls
set interfaces xe-1/3/0 unit 0 family inet address 172.16.25.1/30
set interfaces xe-1/3/0 unit 0 family inet6 address 2001:db8:25:1:1::1/64
set interfaces xe-1/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.10.10.10/32 primary
set interfaces lo0 unit 0 family inet6 address 2001:db8::10:10:10:10/128 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 10.10.10.10
```

```
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/1/0.0 metric 18
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 metric 51
set protocols ospf area 0.0.0.0 interface ge-1/1/1.0 metric 23
set protocols ospf area 0.0.0.0 interface xe-1/3/0.0 metric 52
set protocols ospf area 0.0.0.0 interface ge-1/0/5.0
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/1/0.0 metric 18
set protocols ospf3 area 0.0.0.0 interface xe-0/2/0.0 metric 51
```

```
set protocols ospf3 area 0.0.0.0 interface ge-1/1/1.0 metric 23
set protocols ospf3 area 0.0.0.0 interface xe-1/3/0.0 metric 52
set protocols ospf3 area 0.0.0.0 interface ge-1/0/5.0
```

R1

```
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.2/30
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:10:1:1::2/64
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/5 unit 0 family inet address 172.16.35.1/30
set interfaces ge-0/0/5 unit 0 family inet6 address 2001:db8:35:1:1::1/64
set interfaces ge-0/0/5 unit 0 family mpls
set interfaces xe-0/2/0 unit 0 family inet address 172.16.40.1/30
set interfaces xe-0/2/0 unit 0 family inet6 address 2001:db8:40:1:1::1/64
set interfaces xe-0/2/0 unit 0 family mpls
set interfaces xe-0/3/0 unit 0 family inet address 172.16.45.1/30
set interfaces xe-0/3/0 unit 0 family inet6 address 2001:db8:45:1:1::1/64
set interfaces xe-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.1.1/32 primary
set interfaces lo0 unit 0 family inet6 address 2001:db8::1:1:1:1/128 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.1.1
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
```

```
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols mpls interface ge-0/0/0.0 srlg srlg9
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/5.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-0/3/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/0/5.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/2/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/3/0.0 metric 10
```

R2

```
set interfaces ge-0/0/2 unit 0 family inet address 172.16.35.2/30
set interfaces ge-0/0/2 unit 0 family inet6 address 2001:db8:35:1:1::2/64
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/1/0 unit 0 family inet address 172.16.50.1/30
```

```
set interfaces ge-0/1/0 unit 0 family inet6 address 2001:db8:50:1:1::1/64
set interfaces ge-0/1/0 unit 0 family mpls
set interfaces xe-0/2/1 unit 0 family inet address 172.16.55.1/30
set interfaces xe-0/2/1 unit 0 family inet6 address 2001:db8:55:1:1::1/64
set interfaces xe-0/2/1 unit 0 family mpls
set interfaces ge-1/0/2 unit 0 family inet address 172.16.60.1/30
set interfaces ge-1/0/2 unit 0 family inet6 address 2001:db8:60:1:1::1/64
set interfaces ge-1/0/2 unit 0 family mpls
set interfaces ge-1/0/9 unit 0 family inet address 172.16.65.1/30
set interfaces ge-1/0/9 unit 0 family inet6 address 2001:db8:65:1:1::1/64
set interfaces ge-1/0/9 unit 0 family mpls
set interfaces ge-1/1/5 unit 0 family inet address 172.16.70.1/30
set interfaces ge-1/1/5 unit 0 family inet6 address 2001:db8:70:1:1::1/64
set interfaces ge-1/1/5 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.2.2/32 primary
set interfaces lo0 unit 0 family inet6 address 2001:db8::2:2:2:2/128 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.2.2
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
```

```
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols mpls interface ge-0/1/0.0 srlg srlg1
set protocols mpls interface ge-1/0/9.0 srlg srlg1
set protocols mpls interface ge-1/1/5.0 srlg srlg7
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/1/0.0 link-protection
set protocols ospf area 0.0.0.0 interface xe-0/2/1.0 metric 12
set protocols ospf area 0.0.0.0 interface ge-1/0/2.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-1/0/9.0 metric 12
set protocols ospf area 0.0.0.0 interface ge-1/1/5.0 metric 13
set protocols ospf3 area 0.0.0.0 interface ge-0/0/2.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/1/0.0 link-protection
set protocols ospf3 area 0.0.0.0 interface xe-0/2/1.0 metric 12
set protocols ospf3 area 0.0.0.0 interface ge-1/0/2.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-1/0/9.0 metric 12
set protocols ospf3 area 0.0.0.0 interface ge-1/1/5.0 metric 13
```

R3

```
set interfaces ge-0/0/5 unit 0 family inet address 172.16.50.2/30
set interfaces ge-0/0/5 unit 0 family inet6 address 2001:db8:50:1:1::2/64
set interfaces ge-0/0/5 unit 0 family mpls
```

```
set interfaces xe-0/3/1 unit 0 family inet address 172.16.75.1/30
set interfaces xe-0/3/1 unit 0 family inet6 address 2001:db8:75:1:1::1/64
set interfaces xe-0/3/1 unit 0 family mpls
set interfaces ge-1/0/0 unit 0 family inet address 172.16.80.1/30
set interfaces ge-1/0/0 unit 0 family inet6 address 2001:db8:80:1:1::1/64
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/5 unit 0 family inet address 172.16.200.1/24
set interfaces ge-1/0/5 unit 0 family inet6 address 2001:db8:200:1:1::1/64
set interfaces ge-1/0/6 unit 0 family inet address 172.16.85.1/30
set interfaces ge-1/0/6 unit 0 family inet6 address 2001:db8:85:1:1::1/64
set interfaces ge-1/0/6 unit 0 family mpls
set interfaces xe-1/3/0 unit 0 family inet address 172.16.90.1/30
set interfaces xe-1/3/0 unit 0 family inet6 address 2001:db8:90:1:1::1/64
set interfaces xe-1/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.3.3/32 primary
set interfaces lo0 unit 0 family inet6 address 2001:db8::3:3:3:3/128 primary
set interfaces lo0 unit 0 family mpls
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.3.3
set routing-options forwarding-table export ecmp
set routing-options backup-selection destination 10.1.1.0/30 interface xe-1/3/0.0 admin-group
include-all c2
set routing-options backup-selection destination 10.1.1.0/30 interface all admin-group exclude c3
set routing-options backup-selection destination 10.1.1.0/30 interface all srlg strict
set routing-options backup-selection destination 10.1.1.0/30 interface all protection-type node
set routing-options backup-selection destination 10.1.1.0/30 interface all bandwidth-greater-
equal-primary
set routing-options backup-selection destination 10.1.1.0/30 interface all neighbor preference
172.16.7.7
set routing-options backup-selection destination 10.1.1.0/30 interface all root-metric lowest
set routing-options backup-selection destination 10.1.1.0/30 interface all metric-order root
set routing-options backup-selection destination 172.16.30.0/30 interface all admin-group
```

```
exclude c5
set routing-options backup-selection destination 172.16.30.0/30 interface all srlg strict
set routing-options backup-selection destination 172.16.30.0/30 interface all protection-type
node
set routing-options backup-selection destination 172.16.30.0/30 interface all bandwidth-greater-
equal-primary
set routing-options backup-selection destination 172.16.30.0/30 interface all neighbor
preference 172.16.7.7
set routing-options backup-selection destination 172.16.30.0/30 interface all root-metric lowest
set routing-options backup-selection destination 172.16.30.0/30 interface all metric-order root
set routing-options backup-selection destination 172.16.45.0/30 interface all admin-group
exclude c5
set routing-options backup-selection destination 172.16.45.0/30 interface all srlg strict
set routing-options backup-selection destination 172.16.45.0/30 interface all protection-type
node
set routing-options backup-selection destination 172.16.45.0/30 interface all bandwidth-greater-
equal-primary
set routing-options backup-selection destination 172.16.45.0/30 interface all neighbor
preference 172.16.7.7
set routing-options backup-selection destination 172.16.45.0/30 interface all root-metric lowest
set routing-options backup-selection destination 172.16.45.1/30 interface all metric-order root
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
```

```
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols mpls interface ge-0/0/5.0 admin-group c0
set protocols ospf area 0.0.0.0 interface ge-0/0/5.0 link-protection
set protocols ospf area 0.0.0.0 interface ge-0/0/5.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-0/3/1.0 metric 21
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0 metric 13
set protocols ospf area 0.0.0.0 interface ge-1/0/6.0 metric 15
set protocols ospf area 0.0.0.0 interface xe-1/3/0.0 link-protection
set protocols ospf area 0.0.0.0 interface xe-1/3/0.0 metric 22
set protocols ospf3 area 0.0.0.0 interface ge-0/0/5.0 link-protection
set protocols ospf3 area 0.0.0.0 interface ge-0/0/5.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/3/1.0 metric 21
set protocols ospf3 area 0.0.0.0 interface ge-1/0/0.0 metric 13
set protocols ospf3 area 0.0.0.0 interface ge-1/0/6.0 metric 15
set protocols ospf3 area 0.0.0.0 interface xe-1/3/0.0 link-protection
set protocols ospf3 area 0.0.0.0 interface xe-1/3/0.0 metric 22
set policy-options policy-statement ecmp term 1 then load-balance per-packet
```

R4

```
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
```

```
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.4.4
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols ospf area 0.0.0.0 interface ge-0/1/0.0 metric 18
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-1/3/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-1/1/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-0/3/1.0 metric 21
set protocols ospf3 area 0.0.0.0 interface ge-0/1/0.0 metric 18
```

```
set protocols ospf3 area 0.0.0.0 interface xe-0/2/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-1/3/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-1/1/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/3/1.0 metric 21
```

R5

```
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.5.5
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
```

```
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 metric 51
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/5.0 metric 13
set protocols ospf area 0.0.0.0 interface ge-0/1/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/2/0.0 metric 51
set protocols ospf3 area 0.0.0.0 interface ge-0/0/1.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/0/5.0 metric 13
set protocols ospf3 area 0.0.0.0 interface ge-0/1/0.0 metric 10
```

R6

```
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.6.6
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
```

```
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 25
set protocols mpls admin-groups c26 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
set protocols mpls interface all
set protocols ospf area 0.0.0.0 interface xe-0/3/0.0 metric 52
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 12
set protocols ospf area 0.0.0.0 interface ge-0/0/4.0 metric 15
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface xe-0/3/0.0 metric 52
set protocols ospf3 area 0.0.0.0 interface ge-0/0/0.0 metric 12
set protocols ospf3 area 0.0.0.0 interface ge-0/0/4.0 metric 15
set protocols ospf3 area 0.0.0.0 interface xe-0/2/0.0 metric 10
```

R7

```
set routing-options srlg srlg1 srlg-value 1001
set routing-options srlg srlg2 srlg-value 1002
```

```
set routing-options srlg srlg3 srlg-value 1003
set routing-options srlg srlg4 srlg-value 1004
set routing-options srlg srlg5 srlg-value 1005
set routing-options srlg srlg6 srlg-value 1006
set routing-options srlg srlg7 srlg-value 1007
set routing-options srlg srlg8 srlg-value 1008
set routing-options srlg srlg9 srlg-value 1009
set routing-options srlg srlg10 srlg-value 10010
set routing-options srlg srlg11 srlg-value 10011
set routing-options srlg srlg12 srlg-value 10012
set routing-options router-id 172.16.7.7
set protocols rsvp interface all
set protocols mpls admin-groups c0 0
set protocols mpls admin-groups c1 1
set protocols mpls admin-groups c2 2
set protocols mpls admin-groups c3 3
set protocols mpls admin-groups c4 4
set protocols mpls admin-groups c5 5
set protocols mpls admin-groups c6 6
set protocols mpls admin-groups c7 7
set protocols mpls admin-groups c8 8
set protocols mpls admin-groups c9 9
set protocols mpls admin-groups c10 10
set protocols mpls admin-groups c11 11
set protocols mpls admin-groups c12 12
set protocols mpls admin-groups c13 13
set protocols mpls admin-groups c14 14
set protocols mpls admin-groups c15 15
set protocols mpls admin-groups c16 16
set protocols mpls admin-groups c17 17
set protocols mpls admin-groups c18 18
set protocols mpls admin-groups c19 19
set protocols mpls admin-groups c20 20
set protocols mpls admin-groups c21 21
set protocols mpls admin-groups c22 22
set protocols mpls admin-groups c23 23
set protocols mpls admin-groups c24 24
set protocols mpls admin-groups c25 26
set protocols mpls admin-groups c27 27
set protocols mpls admin-groups c28 28
set protocols mpls admin-groups c29 29
set protocols mpls admin-groups c30 30
set protocols mpls admin-groups c31 31
```

```
set protocols mpls interface all
set protocols mpls interface xe-0/3/0.0 srlg srlg8
set protocols ospf area 0.0.0.0 interface ge-0/1/5.0 metric 23
set protocols ospf area 0.0.0.0 interface xe-0/3/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0 metric 13
set protocols ospf area 0.0.0.0 interface xe-1/3/0.0 metric 22
set protocols ospf area 0.0.0.0 interface xe-1/2/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-0/1/5.0 metric 23
set protocols ospf3 area 0.0.0.0 interface xe-0/3/0.0 metric 10
set protocols ospf3 area 0.0.0.0 interface ge-1/0/0.0 metric 13
set protocols ospf3 area 0.0.0.0 interface xe-1/3/0.0 metric 22
set protocols ospf3 area 0.0.0.0 interface xe-1/2/0.0 metric 10
```

Configuring Device R3

Step-by-Step Procedure

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

To configure Device R3:

1. Configure the interfaces.

```
[edit interfaces]
user@R3# set ge-0/0/5 unit 0 family inet address 172.16.50.2/30
user@R3# set ge-0/0/5 unit 0 family inet6 address 2001:db8:50:1:1::2/64
user@R3# set ge-0/0/5 unit 0 family mpls
user@R3# set xe-0/3/1 unit 0 family inet address 172.16.75.1/30
user@R3# set xe-0/3/1 unit 0 family inet6 address 2001:db8:75:1:1::1/64
user@R3# set xe-0/3/1 unit 0 family mpls
user@R3# set ge-1/0/0 unit 0 family inet address 172.16.80.1/30
user@R3# set ge-1/0/0 unit 0 family inet6 address 2001:db8:80:1:1::1/64
user@R3# set ge-1/0/0 unit 0 family mpls
user@R3# set ge-1/0/5 unit 0 family inet address 172.16.200.1/24
user@R3# set ge-1/0/5 unit 0 family inet6 address 2001:db8:200:1:1::1/64
user@R3# set ge-1/0/6 unit 0 family inet address 172.16.85.1/30
user@R3# set ge-1/0/6 unit 0 family inet6 address 2001:db8:85:1:1::1/64
user@R3# set ge-1/0/6 unit 0 family mpls
user@R3# set xe-1/3/0 unit 0 family inet address 172.16.90.1/30
user@R3# set xe-1/3/0 unit 0 family inet6 address 2001:db8:90:1:1::1/64
```

```
user@R3# set xe-1/3/0 unit 0 family mpls
user@R3# set lo0 unit 0 family inet address 172.16.3.3/32 primary
user@R3# set lo0 unit 0 family inet6 address 2001:db8::3:3:3:3/128 primary
user@R3# set lo0 unit 0 family mpls
```

2. Configure srlg values.

```
[edit routing-options]
user@R3# set srlg srlg1 srlg-value 1001
user@R3# set srlg srlg2 srlg-value 1002
user@R3# set srlg srlg3 srlg-value 1003
user@R3# set srlg srlg4 srlg-value 1004
user@R3# set srlg srlg5 srlg-value 1005
user@R3# set srlg srlg6 srlg-value 1006
user@R3# set srlg srlg7 srlg-value 1007
user@R3# set srlg srlg8 srlg-value 1008
user@R3# set srlg srlg9 srlg-value 1009
user@R3# set srlg srlg10 srlg-value 10010
user@R3# set srlg srlg11 srlg-value 10011
user@R3# set srlg srlg12 srlg-value 10012
```

3. Configure the ID of the router.

```
[edit routing-options]
user@R3# set router-id 172.16.3.3
```

4. Apply the routing policy to all equal-cost multipaths exported from the routing table to the forwarding table.

```
[edit routing-options]
user@R3# set forwarding-table export ecmp
```

5. Configure attributes of the backup selection policy.

```
[edit routing-options backup-selection]
user@R3# set destination 10.1.1.0/30 interface xe-1/3/0.0 admin-group include-all c2
user@R3# set destination 10.1.1.0/30 interface all admin-group exclude c3
user@R3# set destination 10.1.1.0/30 interface all srlg strict
user@R3# set destination 10.1.1.0/30 interface all protection-type node
```

```
user@R3# set destination 10.1.1.0/30 interface all bandwidth-greater-equal-primary
user@R3# set destination 10.1.1.0/30 interface all neighbor preference 172.16.7.7
user@R3# set destination 10.1.1.0/30 interface all root-metric lowest
user@R3# set destination 10.1.1.0/30 interface all metric-order root
user@R3# set destination 172.16.30.0/30 interface all admin-group exclude c5
user@R3# set destination 172.16.30.0/30 interface all srlg strict
user@R3# set destination 172.16.30.0/30 interface all protection-type node
user@R3# set destination 172.16.30.0/30 interface all bandwidth-greater-equal-primary
user@R3# set destination 172.16.30.0/30 interface all neighbor preference 172.16.7.7
user@R3# set destination 172.16.30.0/30 interface all root-metric lowest
user@R3# set destination 172.16.30.0/30 interface all metric-order root
user@R3# set destination 192.168.45.0/30 interface all admin-group exclude c5
user@R3# set destination 192.168.45.0/30 interface all srlg strict
user@R3# set destination 192.168.45.0/30 interface all protection-type node
user@R3# set destination 192.168.45.0/30 interface all bandwidth-greater-equal-primary
user@R3# set destination 192.168.45.0/30 interface all neighbor preference 172.16.7.7
user@R3# set destination 192.168.45.0/30 interface all root-metric lowest
user@R3# set destination 192.168.45.0/30 interface all metric-order root
```

6. Enable RSVP on all the interfaces.

```
[edit protocols]
user@R3# set rsvp interface all
```

7. Configure administrative groups.

```
[edit protocols mpls]
user@R3# set admin-groups c0 0
user@R3# set admin-groups c1 1
user@R3# set admin-groups c2 2
user@R3# set admin-groups c3 3
user@R3# set admin-groups c4 4
user@R3# set admin-groups c5 5
user@R3# set admin-groups c6 6
user@R3# set admin-groups c7 7
user@R3# set admin-groups c8 8
user@R3# set admin-groups c9 9
user@R3# set admin-groups c10 10
user@R3# set admin-groups c11 11
user@R3# set admin-groups c12 12
user@R3# set admin-groups c13 13
```

```
user@R3# set admin-groups c14 14
user@R3# set admin-groups c15 15
user@R3# set admin-groups c16 16
user@R3# set admin-groups c17 17
user@R3# set admin-groups c18 18
user@R3# set admin-groups c19 19
user@R3# set admin-groups c20 20
user@R3# set admin-groups c21 21
user@R3# set admin-groups c22 22
user@R3# set admin-groups c23 23
user@R3# set admin-groups c24 24
user@R3# set admin-groups c25 25
user@R3# set admin-groups c26 26
user@R3# set admin-groups c27 27
user@R3# set admin-groups c28 28
user@R3# set admin-groups c29 29
user@R3# set admin-groups c30 30
user@R3# set admin-groups c31 31
```

8. Enable MPLS on all the interfaces and configure administrative group for an interface.

```
[edit protocols mpls]
user@R3# set interface all
user@R3# set interface ge-0/0/5.0 admin-group c0
```

9. Enable link protection and configure metric values on all the interfaces for an OSPF area.

```
[edit protocols ospf]
user@R3# set area 0.0.0.0 interface ge-0/0/5.0 link-protection
user@R3# set area 0.0.0.0 interface ge-0/0/5.0 metric 10
user@R3# set area 0.0.0.0 interface xe-0/3/1.0 metric 21
user@R3# set area 0.0.0.0 interface ge-1/0/0.0 metric 13
user@R3# set area 0.0.0.0 interface ge-1/0/6.0 metric 15
user@R3# set area 0.0.0.0 interface xe-1/3/0.0 link-protection
user@R3# set area 0.0.0.0 interface xe-1/3/0.0 metric 22
```

10. Enable link protection and configure metric values on all the interfaces for an OSPF3 area.

```
[edit protocols ospf3]
user@R3# set area 0.0.0.0 interface ge-0/0/5.0 link-protection
```

```
user@R3# set area 0.0.0.0 interface ge-0/0/5.0 metric 10
user@R3# set area 0.0.0.0 interface xe-0/3/1.0 metric 21
user@R3# set area 0.0.0.0 interface ge-1/0/0.0 metric 13
user@R3# set area 0.0.0.0 interface ge-1/0/6.0 metric 15
user@R3# set area 0.0.0.0 interface xe-1/3/0.0 link-protection
user@R3# set area 0.0.0.0 interface xe-1/3/0.0 metric 22
```

11. Configure the routing policy.

```
[edit policy-options]
user@R3# set policy-statement ecmp term 1 then load-balance per-packet
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
ge-0/0/5 {
    unit 0 {
        family inet {
            address 192.168.50.2/30;
        }
        family inet6 {
            address 2001:db8:50:1:1::2/64;
        }
        family mpls;
    }
}
xe-0/3/1 {
    unit 0 {
        family inet {
            address 192.168.75.1/30;
        }
        family inet6 {
            address 2001:db8:75:1:1::1/64;
        }
        family mpls;
    }
```

```
}
ge-1/0/0 {
    unit 0 {
        family inet {
            address 192.168.80.1/30;
        }
        family inet6 {
            address 2001:db8:80:1:1::1/64;
        }
        family mpls;
    }
}
ge-1/0/5 {
    unit 0 {
        family inet {
            address 172.16.200.1/24;
        }
        family inet6 {
            address 2001:db8:200:1:1::1/64;
        }
    }
}
ge-1/0/6 {
    unit 0 {
        family inet {
            address 192.168.85.1/30;
        }
        family inet6 {
            address 2001:db8:85:1:1::1/64;
        }
        family mpls;
    }
}
xe-1/3/0 {
    unit 0 {
        family inet {
            address 192.168.90.1/30;
        }
        family inet6 {
            address 2001:db8:90:1:1::1/64;
        }
        family mpls;
    }
```

```
}
lo0 {
    unit 0 {
        family inet {
            address 172.16.3.3/32 {
                primary;
            }
        }
        family inet6 {
            address 2001:db8:3:3:3:3/128 {
                primary;
            }
        }
        family mpls;
    }
}
```

```
user@R3# show protocols
rsvp {
    interface all;
}
mpls {
    admin-groups {
        c0 0;
        c1 1;
        c2 2;
        c3 3;
        c4 4;
        c5 5;
        c6 6;
        c7 7;
        c8 8;
        c9 9;
        c10 10;
        c11 11;
        c12 12;
        c13 13;
        c14 14;
        c15 15;
        c16 16;
        c17 17;
```

```
        c18 18;
        c19 19;
        c20 20;
        c21 21;
        c22 22;
        c23 23;
        c24 24;
        c25 25;
        c26 26;
        c27 27;
        c28 28;
        c29 29;
        c30 30;
        c31 31;
    }
    interface all;
    interface ge-0/0/5.0 {
        admin-group c0;
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/5.0 {
            link-protection;
            metric 10;
        }
        interface xe-0/3/1.0 {
            metric 21;
        }
        interface ge-1/0/0.0 {
            metric 13;
        }
        interface ge-1/0/6.0 {
            metric 15;
        }
        interface xe-1/3/0.0 {
            link-protection;
            metric 22;
        }
    }
}
ospf3 {
    area 0.0.0.0 {
```

```
        interface ge-0/0/5.0 {
            link-protection;
            metric 10;
        }
        interface xe-0/3/1.0 {
            metric 21;
        }
        interface ge-1/0/0.0 {
            metric 13;
        }
        interface ge-1/0/6.0 {
            metric 15;
        }
        interface xe-1/3/0.0 {
            link-protection;
            metric 22;
        }
    }
}
```

```
user@R3# show routing-options
srlg {
    srlg1 srlg-value 1001;
    srlg2 srlg-value 1002;
    srlg3 srlg-value 1003;
    srlg4 srlg-value 1004;
    srlg5 srlg-value 1005;
    srlg6 srlg-value 1006;
    srlg7 srlg-value 1007;
    srlg8 srlg-value 1008;
    srlg9 srlg-value 1009;
    srlg10 srlg-value 10010;
    srlg11 srlg-value 10011;
    srlg12 srlg-value 10012;
}
router-id 172.16.3.3;
forwarding-table {
    export ecmp;
}
backup-selection {
    destination 10.1.1.0/30 {
```

```
        interface xe-1/3/0.0 {
            admin-group {
                include-all c2;
            }
        }
        interface all {
            admin-group {
                exclude c3;
            }
            srlg strict;
            protection-type node;
            bandwidth-greater-equal-primary;
            node {
                preference 172.16.7.7;
            }
            root-metric lowest;
            metric-order root;
        }
    }
    destination 172.16.30.0/30 {
        interface all {
            admin-group {
                exclude c5;
            }
            srlg strict;
            protection-type node;
            bandwidth-greater-equal-primary;
            node {
                preference 172.16.7.7;
            }
            root-metric lowest;
            metric-order root;
        }
    }
    destination 192.168.45.0/30 {
        interface all {
            admin-group {
                exclude c5;
            }
            srlg strict;
            protection-type node;
            bandwidth-greater-equal-primary;
            node {
```

```
                preference 172.16.7.7;
            }
            root-metric lowest;
            metric-order root;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes | **248**
- Verifying the OSPF Route | **252**
- Verifying the OSPF3 Route | **253**
- Verifying the Backup Selection Policy for Device R3 | **253**

Confirm that the configuration is working properly.

Verifying the Routes

Purpose

Verify that the expected routes are learned.

Action

From operational mode, run the `show route` command for the routing table.

```
user@R3> show route
inet.0: 48 destinations, 48 routes (48 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.3.3/32         *[Direct/0] 02:22:27
                       > via lo0.0
10.4.0.0/16          *[Static/5] 02:22:57
```

```
                    > to 10.92.31.254 via fxp0.0
10.5.0.0/16         *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.6.128.0/17       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.9.0.0/16         *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.10.0.0/16        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.13.4.0/23        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.13.10.0/23       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.82.0.0/15        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.84.0.0/16        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.85.12.0/22       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.92.0.0/16        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.92.16.0/20       *[Direct/0] 02:22:57
                    > via fxp0.0
10.92.24.195/32     *[Local/0] 02:22:57
                      Local via fxp0.0
10.94.0.0/16        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.99.0.0/16        *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.102.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.150.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.155.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.157.64.0/19      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.160.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.204.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.205.0.0/16       *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
```

```
10.206.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.207.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.209.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.212.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.213.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.214.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.215.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.216.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.218.13.0/24     *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.218.14.0/24     *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.218.16.0/20     *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.218.32.0/20     *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
10.227.0.0/16      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
172.16.50.0/30       *[Direct/0] 02:19:55
                    > via ge-0/0/5.0
172.16.50.2/32       *[Local/0] 02:19:58
                      Local via ge-0/0/5.0
172.16.75.0/30       *[Direct/0] 02:19:55
                    > via xe-0/3/1.0
172.16.75.1/32       *[Local/0] 02:19:57
                      Local via xe-0/3/1.0
172.16.24.195/32   *[Direct/0] 02:22:57
                    > via lo0.0
172.16.0.0/12      *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
192.168.0.0/16     *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
192.168.102.0/23   *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
192.168.136.0/24    *[Static/5] 02:22:57
```

```
                    > to 10.92.31.254 via fxp0.0
192.168.136.192/32  *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
192.168.137.0/24    *[Static/5] 02:22:57
                    > to 10.92.31.254 via fxp0.0
192.168.233.5/32       *[OSPF/10] 00:16:55, metric 1
                       MultiRecv

iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

47.0005.80ff.f800.0000.0108.0001.1280.9202.4195/152
                   *[Direct/0] 02:22:57
                    > via lo0.0

mpls.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 00:16:55, metric 1
                      Receive
1                  *[MPLS/0] 00:16:55, metric 1
                      Receive
2                  *[MPLS/0] 00:16:55, metric 1
                      Receive
13                 *[MPLS/0] 00:16:55, metric 1
                      Receive

inet6.0: 10 destinations, 11 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8:50:1:1::/64   *[Direct/0] 02:19:44
                    > via ge-0/0/5.0
2001:db8:50:1:1::2/128 *[Local/0] 02:19:58
                       Local via ge-0/0/5.0
2001:db8:75:1:1::/64   *[Direct/0] 02:19:44
                    > via xe-0/3/1.0
2001:db8:75:1:1::1/128 *[Local/0] 02:19:57
                       Local via xe-0/3/1.0
2001:db8::3:3:3:3/128  *[Direct/0] 02:22:27
                    > via lo0.0
2001:db8::128:92:24:195/128
                   *[Direct/0] 02:22:57
                    > via lo0.0
```

```
fe80::/64          *[Direct/0] 02:19:44
                    > via ge-0/0/5.0
                    [Direct/0] 02:19:43
                    > via xe-0/3/1.0
fe80::205:86ff:fe00:ed05/128
                   *[Local/0] 02:19:58
                      Local via ge-0/0/5.0
fe80::205:86ff:fe00:ed3d/128
                   *[Local/0] 02:19:57
                      Local via xe-0/3/1.0
fe80::5668:a50f:fcc1:3ca2/128
                   *[Direct/0] 02:22:57
                    > via lo0.0
```

Meaning

The output shows all Device R3 routes.

Verifying the OSPF Route

Purpose

Verify the routing table of OSPF.

Action

From operational mode, run the `show ospf route detail` command for Device R3.

```
user@R3> show ospf route detail
Topology default Route Table:

Prefix             Path  Route      NH       Metric NextHop       Nexthop
                   Type  Type       Type            Interface     Address/LSP
172.16.50.0/30        Intra Network    IP           10 ge-0/0/5.0
  area 0.0.0.0, origin 172.16.3.3, priority low
172.16.75.0/30        Intra Network    IP           21 xe-0/3/1.0
  area 0.0.0.0, origin 172.16.3.3, priority low
```

Meaning

The output displays the routing table of OSPF routers.

Verifying the OSPF3 Route

Purpose

Verify the routing table of OSPF3.

Action

From operational mode, run the `show ospf3 route detail` command for Device R3.

```
user@R3> show ospf3 route detail

Prefix                                  Path  Route      NH   Metric
                                        Type  Type       Type
2001:db8:50:1:1::/64                        Intra Network    IP   10
  NH-interface ge-0/0/5.0
  Area 0.0.0.0, Origin 172.16.3.3, Priority low
2001:db8:75:1:1::/64                        Intra Network    IP   21
  NH-interface xe-0/3/1.0
  Area 0.0.0.0, Origin 172.16.3.3, Priority low
```

Meaning

The output displays the routing table of OSPF3 routers.

Verifying the Backup Selection Policy for Device R3

Purpose

Verify the backup selection policy for Device R3.

Action

From operational mode, run the `show backup-selection` command for Device R3.

```
user@R3> show backup-selection
Prefix: 10.1.1.0/30
  Interface: all
    Admin-group exclude: c3
    Neighbor preference: 172.16.7.7
    Protection Type: Node, Downstream Paths Only: Disabled, SRLG: Strict, B/w >= Primary:
Enabled, Root-metric: lowest, Dest-metric: lowest
    Metric Evaluation Order: Root-metric, Dest-metric
    Policy Evaluation Order: Admin-group, SRLG, Bandwidth, Protection, node, Metric
  Interface: xe-1/3/0.0
    Admin-group include-all: c2
    Protection Type: Link, Downstream Paths Only: Disabled, SRLG: Loose, B/w >= Primary:
Disabled, Root-metric: lowest, Dest-metric: lowest
    Metric Evaluation Order: Dest-metric, Root-metric
    Policy Evaluation Order: Admin-group, SRLG, Bandwidth, Protection, node, Metric Prefix:
172.16.30.0/30
  Interface: all
    Admin-group exclude: c5
    Neighbor preference: 172.16.7.7
    Protection Type: Node, Downstream Paths Only: Disabled, SRLG: Strict, B/w >= Primary:
Enabled, Root-metric: lowest, Dest-metric: lowest
    Metric Evaluation Order: Root-metric, Dest-metric
    Policy Evaluation Order: Admin-group, SRLG, Bandwidth, Protection, node, Metric
Prefix: 172.16.45.0/30
  Interface: all
    Admin-group exclude: c5
    Neighbor preference: 172.16.7.7
    Protection Type: Node, Downstream Paths Only: Disabled, SRLG: Strict, B/w >= Primary:
Enabled, Root-metric: lowest, Dest-metric: lowest
    Metric Evaluation Order: Root-metric, Dest-metric
    Policy Evaluation Order: Admin-group, SRLG, Bandwidth, Protection, node, Metric
```

Meaning

The output displays the configured policies per prefix per primary next-hop interface.

RELATED DOCUMENTATION

- backup-selection (Protocols IS-IS)

CHAPTER 3

Evaluating Complex Cases Using Policy Chains and Subroutines

IN THIS CHAPTER

- Understanding How a Routing Policy Chain Is Evaluated | 256
- Example: Configuring Policy Chains and Route Filters | 258
- Example: Using Firewall Filter Chains | 275
- Understanding Policy Subroutines in Routing Policy Match Conditions | 283
- How a Routing Policy Subroutine Is Evaluated | 287
- Example: Configuring a Policy Subroutine | 289

Understanding How a Routing Policy Chain Is Evaluated

Figure 15 on page 257 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:

> **NOTE**: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

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.

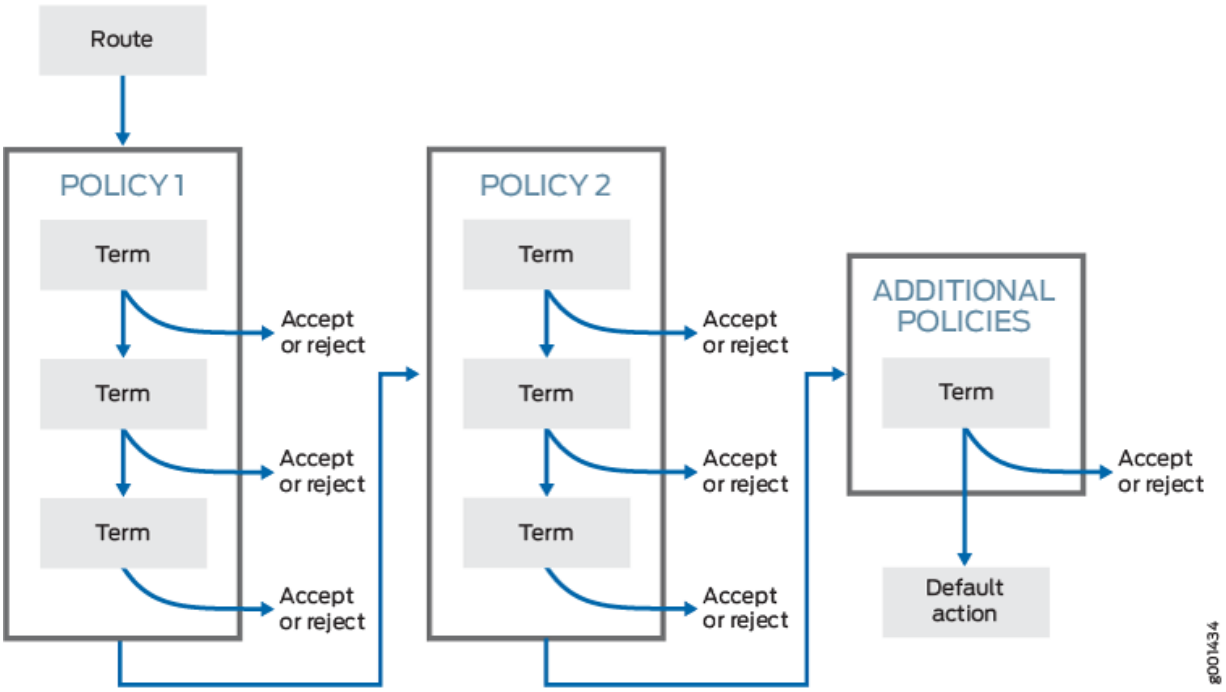
2. The route is evaluated against the second term in the first routing policy. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of

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.

Figure 15: Routing Policy Chain Evaluation



RELATED DOCUMENTATION

Default Routing Policies | 40

Example: Configuring Policy Chains and Route Filters | 258

Example: Configuring Policy Chains and Route Filters

IN THIS SECTION

- Requirements | 258
- Overview | 258
- Configuration | 261
- Verification | 270

A *policy chain* is the application of multiple policies within a specific section of the configuration. A *route filter* is a collection of match prefixes.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 260

An example of a policy chain applied to BGP is as follows:

```
user@R1# show protocols bgp
group int {
    type internal;
    local-address 192.168.0.1;
    export [ adv-statics adv-large-aggregates adv-small-aggregates ];
    neighbor 192.168.0.2;
    neighbor 192.168.0.3;
}
```

The adv-statics, adv-large-aggregates, and adv-small-aggregates policies, in addition to the default BGP policy, make up the policy chain applied to the BGP peers of Device R1. Two of the policies demonstrate route filters with different match types. The other policy matches all static routes, so no route filter is needed.

```
user@R1# show policy-options
policy-statement adv-large-aggregates {
    term between-16-and-18 {
        from {
            protocol aggregate;
            route-filter 172.16.0.0/16 upto /18;
        }
        then accept;
    }
}
policy-statement adv-small-aggregates {
    term between-19-and-24 {
        from {
            protocol aggregate;
            route-filter 172.16.0.0/16 prefix-length-range /19-/24;
        }
        then accept;
    }
}
policy-statement adv-statics {
    term statics {
        from protocol static;
        then accept;
    }
}
```

Optionally, you can convert this policy chain into a single multiterm policy for the internal BGP (IBGP) peers. If you do this, one of the advantages of a policy chain is lost—the ability to reuse policies for different purposes.

Figure 16 on page 260 displays Device R1 in AS 64510 with its IBGP peers, Device R2 and Device R3. Device R1 also has external BGP (EBGP) connections to Device R4 in AS 64511 and Device R5 in AS 64512. The current administrative policy within AS 64510 is to send the customer static routes only to other IBGP peers. Any EBGP peer providing transit service only receives aggregate routes with mask lengths smaller than 18 bits. Any EBGP peer providing peering services receives all customer routes and all aggregates whose mask length is larger than 19 bits. Each portion of these administrative policies is configured in a separate routing policy within the [edit policy-opitons] configuration hierarchy. These

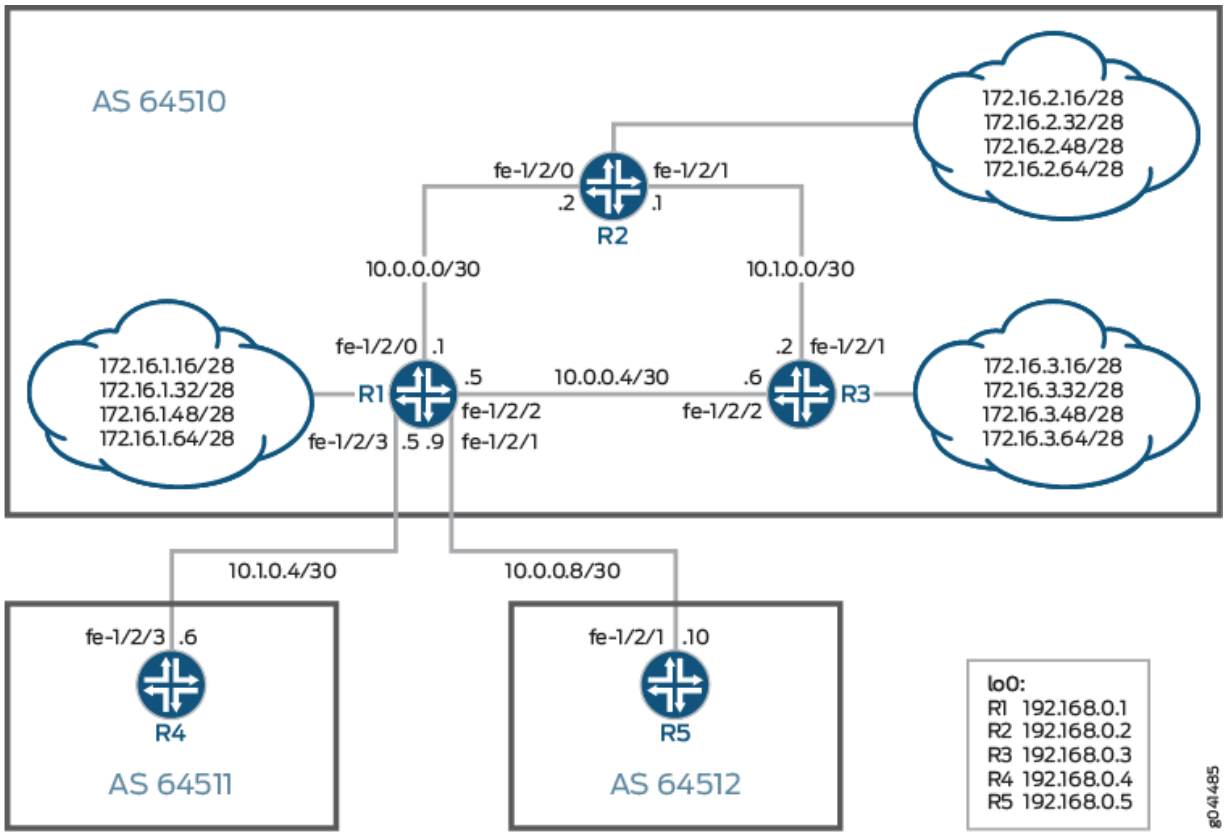
policies provide the administrators of AS 64510 with multiple configuration options for advertising routes to peers.

Device R4 is providing transit service to AS 64510, which allows the AS to advertise its assigned routing space to the Internet. On the other hand, the peering service provided by Device R5 allows AS 64510 to route traffic directly between the autonomous systems (ASs) for all customer routes.

Topology

Figure 16 on page 260 shows the sample network.

Figure 16: BGP Topology for Policy Chains



"CLI Quick Configuration" on page 261 shows the configuration for all of the devices in Figure 16 on page 260.

The section "Procedure" on page 264 describes the steps on Device R1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 261
- Procedure | 264

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 description to_R2
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/2 unit 0 description to_R3
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.5/30
set interfaces fe-1/2/3 unit 0 description to_R4
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.5/30
set interfaces fe-1/2/1 unit 0 description to_R5
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.10/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.1
set protocols bgp group int export adv-statics
set protocols bgp group int export adv-large-aggregates
set protocols bgp group int export adv-small-aggregates
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group to_64511 type external
set protocols bgp group to_64511 export adv-large-aggregates
set protocols bgp group to_64511 neighbor 10.1.0.6 peer-as 64511
set protocols bgp group to_64512 type external
set protocols bgp group to_64512 export adv-small-aggregates
set protocols bgp group to_64512 export adv-statics
set protocols bgp group to_64512 neighbor 10.0.0.9 peer-as 64512
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
```

```
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement adv-large-aggregates term between-16-and-18 from protocol
aggregate
set policy-options policy-statement adv-large-aggregates term between-16-and-18 from route-
filter 172.16.0.0/16 upto /18
set policy-options policy-statement adv-large-aggregates term between-16-and-18 then accept
set policy-options policy-statement adv-small-aggregates term between-19-and-24 from protocol
aggregate
set policy-options policy-statement adv-small-aggregates term between-19-and-24 from route-
filter 172.16.0.0/16 prefix-length-range /19-/24
set policy-options policy-statement adv-small-aggregates term between-19-and-24 then accept
set policy-options policy-statement adv-statics term statics from protocol static
set policy-options policy-statement adv-statics term statics then accept
set routing-options static route 172.16.1.16/28 discard
set routing-options static route 172.16.1.32/28 discard
set routing-options static route 172.16.1.48/28 discard
set routing-options static route 172.16.1.64/28 discard
set routing-options aggregate route 172.16.0.0/16
set routing-options aggregate route 172.16.1.0/24
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device R2

```
set interfaces fe-1/2/0 unit 0 description to_R1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 description to_R3
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int neighbor 192.168.0.1 export send-static-aggregate
set protocols bgp group int neighbor 192.168.0.3
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static-aggregate term 1 from protocol static
set policy-options policy-statement send-static-aggregate term 1 from protocol aggregate
set policy-options policy-statement send-static-aggregate term 1 then accept
set routing-options static route 172.16.2.16/28 discard
set routing-options static route 172.16.2.32/28 discard
```

```
set routing-options static route 172.16.2.48/28 discard
set routing-options static route 172.16.2.64/28 discard
set routing-options aggregate route 172.16.2.0/24
set routing-options aggregate route 172.16.0.0/16
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64510
```

Device R3

```
set interfaces fe-1/2/1 unit 0 description to_R2
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces fe-1/2/2 unit 0 description to_R1
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int neighbor 192.168.0.1 export send-static-aggregate
set protocols bgp group int neighbor 192.168.0.2
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static-aggregate from protocol static
set policy-options policy-statement send-static-aggregate from protocol aggregate
set policy-options policy-statement send-static-aggregate then accept
set routing-options static route 172.16.3.16/28 discard
set routing-options static route 172.16.3.32/28 discard
set routing-options static route 172.16.3.48/28 discard
set routing-options static route 172.16.3.64/28 discard
set routing-options aggregate route 172.16.0.0/16
set routing-options aggregate route 172.16.3.0/24
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64510
```

Device R4

```
set interfaces fe-1/2/3 unit 0 description to_R1
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64510
```

```
set protocols bgp group ext neighbor 10.1.0.5
set routing-options autonomous-system 64511
```

Device R5

```
set interfaces fe-1/2/1 unit 0 description to_R1
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.9/30
set interfaces lo0 unit 0 family inet address 192.168.0.5/32
set protocols bgp group ext type external
set protocols bgp group ext neighbor 10.0.0.10 peer-as 64510
set routing-options autonomous-system 64512
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 0 description to_R2
user@R1# set fe-1/2/0 unit 0 family inet address 10.0.0.1/30
user@R1# set fe-1/2/2 unit 0 description to_R3
user@R1# set fe-1/2/2 unit 0 family inet address 10.0.0.5/30
user@R1# set fe-1/2/3 unit 0 description to_R4
user@R1# set fe-1/2/3 unit 0 family inet address 10.1.0.5/30
user@R1# set fe-1/2/1 unit 0 description to_R5
user@R1# set fe-1/2/1 unit 0 family inet address 10.0.0.10/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure the IBGP connections to Device R2 and Device R3.

```
[edit protocols bgp group int]
user@R1# set type internal
user@R1# set local-address 192.168.0.1
```

```
user@R1# set neighbor 192.168.0.2
user@R1# set neighbor 192.168.0.3
```

3. Apply the export policies for the internal peers.

```
[edit protocols bgp group int]
user@R1# set export adv-statics
user@R1# set export adv-large-aggregates
user@R1# set export adv-small-aggregates
```

4. Configure the EBGP connection to Device R4.

```
[edit protocols bgp group to_64511]
user@R1# set type external
user@R1# set neighbor 10.1.0.6 peer-as 64511
```

5. Apply the export policy for Device R4.

```
[edit protocols bgp group to_64511]
user@R1# set export adv-large-aggregates
```

6. Configure the EBGP connection to Device R5.

```
[edit protocols bgp group to_64512]
user@R1# set type external
user@R1# set neighbor 10.0.0.9 peer-as 64512
```

7. Apply the export policies for Device R5.

```
[edit protocols bgp group to_64512]
user@R1# set export adv-small-aggregates
user@R1# set export adv-statics
```

8. Configure OSPF connections to Device R2 and Device R3.

```
[edit protocols ospf area 0.0.0.0]
user@R1# set interface fe-1/2/0.0
```

```
user@R1# set interface fe-1/2/2.0
user@R1# set interface lo0.0 passive
```

9. Configure the routing policies.

```
[edit policy-options policy-statement adv-large-aggregates term between-16-and-18]
user@R1# set from protocol aggregate
user@R1# set from route-filter 172.16.0.0/16 upto /18
user@R1# set then accept
[edit policy-options policy-statement adv-small-aggregates term between-19-and-24]
user@R1# set from protocol aggregate
user@R1# set from route-filter 172.16.0.0/16 prefix-length-range /19-/24
user@R1# set then accept
[edit policy-options policy-statement adv-statics term statics]
user@R1# set from protocol static
user@R1# set then accept
```

10. Configure the static and aggregate routes.

```
[edit routing-options static]
user@R1# set route 172.16.1.16/28 discard
user@R1# set route 172.16.1.32/28 discard
user@R1# set route 172.16.1.48/28 discard
user@R1# set route 172.16.1.64/28 discard
[edit routing-options aggregate]
user@R1# set route 172.16.0.0/16
user@R1# set route 172.16.1.0/24
```

11. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R1# set router-id 192.168.0.1
user@R1# set autonomous-system 64510
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
fe-1/2/0 {
    unit 0 {
        description to_R2;
        family inet {
            address 10.0.0.1/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        description to_R3;
        family inet {
            address 10.0.0.5/30;
        }
    }
}
fe-1/2/3 {
    unit 0 {
        description to_R4;
        family inet {
            address 10.1.0.5/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        description to_R5;
        family inet {
            address 10.0.0.10/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
```

```
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.1;
        export [ adv-statics adv-large-aggregates adv-small-aggregates ];
        neighbor 192.168.0.2;
        neighbor 192.168.0.3;
    }
    group to_64511 {
        type external;
        export adv-large-aggregates;
        neighbor 10.1.0.6 {
            peer-as 64511;
        }
    }
    group to_64512 {
        type external;
        export [ adv-small-aggregates adv-statics ];
        neighbor 10.0.0.9 {
            peer-as 64512;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/0.0;
        interface fe-1/2/2.0;
        interface lo0.0 {
            passive;
        }
```

```
    }
}
```

```
user@R1# show policy-options
policy-statement adv-large-aggregates {
    term between-16-and-18 {
        from {
            protocol aggregate;
            route-filter 172.16.0.0/16 upto /18;
        }
        then accept;
    }
}
policy-statement adv-small-aggregates {
    term between-19-and-24 {
        from {
            protocol aggregate;
            route-filter 172.16.0.0/16 prefix-length-range /19-/24;
        }
        then accept;
    }
}
policy-statement adv-statics {
    term statics {
        from protocol static;
        then accept;
    }
}
```

```
user@R1# show routing-options
static {
    route 172.16.1.16/28 discard;
    route 172.16.1.32/28 discard;
    route 172.16.1.48/28 discard;
    route 172.16.1.64/28 discard;
}
aggregate {
    route 172.16.0.0/16;
    route 172.16.1.0/24;
}
```

```
router-id 192.168.0.1;
autonomous-system 64510;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Route Advertisement to Device R4 | **270**
- Checking Where the Longer Routes Are Originating | **271**
- Blocking the More Specific Routes | **272**
- Verifying the Route Advertisement to Device R5 | **272**

Confirm that the configuration is working properly.

Verifying the Route Advertisement to Device R4

Purpose

On Device R1, make sure that the customer routes are advertised to Device R4.

Action

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.0.0/16           Self                                    I
* 172.16.2.0/24           Self                                    I
* 172.16.2.16/28          Self                                    I
* 172.16.2.32/28          Self                                    I
* 172.16.2.48/28          Self                                    I
* 172.16.2.64/28          Self                                    I
* 172.16.3.0/24           Self                                    I
* 172.16.3.16/28          Self                                    I
* 172.16.3.32/28          Self                                    I
```

```
* 172.16.3.48/28          Self                                  I
* 172.16.3.64/28          Self                                  I
```

Meaning

The `adv-large-aggregates` policy is applied to the peering session with Device R4 to advertise the aggregate routes with a subnet mask length between 16 and 18 bits. The 172.16.0.0/16 aggregate route is being sent as defined by the administrative policy, but a number of other routes with larger subnet masks are also being sent to Device R4.

Checking Where the Longer Routes Are Originating

Purpose

On Device R1, find where the other routes are coming from.

Action

```
user@R1> show route 172.16.3.16/28

inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.3.16/28     *[BGP/170] 20:16:00, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 10.0.0.6 via fe-1/2/2.0
```

Meaning

Device R1 has learned this route through its BGP session with Device R3. Because it is an active BGP route, it is automatically advertised by the BGP default policy. Remember that the default policy is always applied to the end of every policy chain. What is needed is a policy to block the more specific routes from being advertised.

Blocking the More Specific Routes

Purpose

Create a policy called `not-larger-than-18` that rejects all routes within the 172.16.0.0 /16 address space that have a subnet mask length greater than or equal to 19 bits. This ensures that all aggregates with a mask between 16 and 18 bits are advertised, thus accomplishing the goal of the administrative policy.

Action

1. On Device R1, configure the `not-larger-than-18` policy.

```
[edit policy-options policy-statement not-larger-than-18 term reject-greater-than-18-bits]
user@R1# set from route-filter 172.16.0.0/16 prefix-length-range /19-/32
user@R1# set then reject
```

2. On Device R1, apply the policy to the peering session with Device R4.

```
[edit  protocols bgp group to_64511]
user@R1# set export not-larger-than-18
user@R1# commit
```

3. On Device R1, check which routes are advertised to Device R4.

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.0.0/16           Self                                    I
```

Meaning

The policy chain is working correctly. Only the 172.16.0.0 /16 route is advertised to Device R4.

Verifying the Route Advertisement to Device R5

Purpose

On Device R1, make sure that the customer routes are advertised to Device R5.

Device R5 is Device R1's EBGP peer in AS 64512. The administrative policy states that this peer receives only aggregate routes larger than 18 bits in length and all customer routes. In anticipation of encountering a problem similar to the problem on Device R4, you can create a policy called `not-smaller-than-18` that rejects all aggregates with mask lengths between 16 and 18 bits.

Action

1. On Device R2, configure an aggregate route for 172.16.128.0/17.

```
[edit routing-options aggregate]
user@R2# set route 172.16.128.0/17 discard
user@R2# commit
```

2. On Device R1, check which routes are advertised to Device R5.

```
user@R1> show route advertising-protocol bgp 10.0.0.9


inet.0: 30 destinations, 32 routes (30 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.0/24           Self                                    I
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
* 172.16.1.64/28          Self                                    I
* 172.16.2.0/24           Self                                    I
* 172.16.2.16/28          Self                                    I
* 172.16.2.32/28          Self                                    I
* 172.16.2.48/28          Self                                    I
* 172.16.2.64/28          Self                                    I
* 172.16.3.0/24           Self                                    I
* 172.16.3.16/28          Self                                    I
* 172.16.3.32/28          Self                                    I
* 172.16.3.48/28          Self                                    I
* 172.16.3.64/28          Self                                    I
* 172.16.128.0/17         Self                                    I
```

The aggregate route 172.16.128.0/17 is advertised, in violation of the administrative policy

3. On Device R1, configure the `not-smaller-than-18` policy.

```
[edit policy-options policy-statement not-smaller-than-18 term reject-less-than-18-bits]
user@R1# set from protocol aggregate
user@R1# set from route-filter 172.16.0.0/16 upto /18
user@R1# set then reject
```

4. On Device R1, apply the policy to the peering session with Device R5.

```
[edit  protocols bgp group to_64512]
user@R1# set export not-smaller-than-18
user@R1# commit
```

5. On Device R1, check which routes are advertised to Device R5.

```
user@R1> show route advertising-protocol bgp 10.0.0.9

inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.0/24           Self                                    I
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
* 172.16.1.64/28          Self                                    I
* 172.16.2.0/24           Self                                    I
* 172.16.2.16/28          Self                                    I
* 172.16.2.32/28          Self                                    I
* 172.16.2.48/28          Self                                    I
* 172.16.2.64/28          Self                                    I
* 172.16.3.0/24           Self                                    I
* 172.16.3.16/28          Self                                    I
* 172.16.3.32/28          Self                                    I
* 172.16.3.48/28          Self                                    I
* 172.16.3.64/28          Self                                    I
```

Meaning

The policy chain is working correctly. Only aggregate routes larger than 18 bits in length and all customer routes are advertised to Device R5.

RELATED DOCUMENTATION

Understanding Route Filters for Use in Routing Policy Match Conditions | 303

Route Filter Match Conditions | 72

Example: Configuring Routing Policy Prefix Lists | 396

Example: Configuring a Policy Subroutine | 289

Example: Using Firewall Filter Chains

IN THIS SECTION

- Requirements | 275
- Overview | 276
- Configuration | 277
- Verification | 282

This example shows the use of firewall filter chains. Firewall filters filter1, filter2, and filter3, are applied to interface ge-0/1/1.0 using the `input-chain` and the `output-chain` configuration statements.

Requirements

Before you begin:

- You should have a MX Series router with MPCs and running Junos release 18.4R1 or later.

  If you are using PTX10001-36MR, PTX10004, PTX10008, or PTX10016 routers for this feature, install Junos OS Evolved Release 21.4R1.

- The router should be configured for IP version 4 (IPv4) protocol (`family inet`) and configured the logical interface with an interface address. All other initial router configurations should be complete, with basic IPv4 connectivity between the devices confirmed.

- The traffic you send should be compatible with the firewall filter rules so the rules you configure can match the test traffic you send.

Overview

IN THIS SECTION

- Topology | 277

This examples shows how to chain multiple firewall filters for both ingress and egress so they can be applied to a given interface and evaluated in sequence. The order of execution occurs in the same order as the chain, from left to right.

Using filter chains (as opposed to input-list filter) has the advantage of allowing multiple levels of filtering, such as using an initial filter to perform generic classification (such as QoS), and then one or more subsequent filters for additional refinement (such as security) because they avoid the inherit conflict that can come when IP addresses used in the evaluation overlap.

Starting from Junos OS Evolved Release 21.4R1, you can use firewall filter chains on PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers.

You can apply the filter chain as follows:

```
set interfaces interface-name unit unit family inet filter input-chain [filter1 filter2 filter3];
```

```
set interfaces interface-name unit unit family inet filter output-chain [filter1 filter2 filter3];
```

On PTX Evo platforms, the feature has the following limitations:

- You can configure only the first filter in a chain of filters as interface specific. On MX Series routers, you can configure all filters in a chain of filters as interface specific.
- You cannot configure the same filters as part of a regular CLI filter and chain filters on the same interface specific bind point. On such interface specific bind points, replace the existing CLI filter with filter chains or vice-versa and commit them separately, to avoid an error.
- You cannot configure chain filters along with “family ANY” and interface-policers on the same bind point.
- On loopback interfaces, output chain filters are not supported.
- On loopback interfaces, you cannot configure both input CLI regular filter and chain filters.
- For IRB interfaces, you cannot configure both regular CLI interface-specific filter and filter chains.
- For Layer 2 SP style output, you cannot configure both regular CLI interface specific filter and chain filters.

- Filters such as `fast-lookup-filter` are not supported as part of CLI chain filters.
- CLI filters chains are not supported for Urpf-fail-filters.
- As egress filters for MPLS family are supported as `fast-lookup-filter` only and chain filters do not support fast-lookup-filters, relevant commit check will be provided while configuring the family MPLS egress chain filters.

Topology

In this example, you configure multiple firewall filters and then apply them in sequence by chaining them to a given interface. This example uses `ge-0/1/1.0` configured with the IP address 172.16.1.1/30 for both the input and output chain. If a packet does not match any of the filters in the chain list, the packet is dropped.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 277
- Configure IPv4 Firewall Filters | 278
- Apply the Chain of Input Filters | 279
- Confirm and Commit Your Candidate Configuration | 280

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level. The filter names used here are **filter1**, and so on, while the term names are **t1_f1** (term1, using filter1), and so on.

```
set firewall family inet filter filter1 term t1_f1 from protocol tcp
set firewall family inet filter filter1 term t1_f1 then count f1_t1_cnt
set firewall family inet filter filter1 term t2_f1 from precedence 7
set firewall family inet filter filter1 term t2_f1 then count f1_t2_cnt
set firewall family inet filter filter1 term t2_f1 then accept
set firewall family inet filter filter2 term t1_f2 from dscp 0
```

```
set firewall family inet filter filter2 term t1_f2 then count f2_t1_cnt
set firewall family inet filter filter2 term t2_f2 from source-port 1020
set firewall family inet filter filter2 term t2_f2 then count f2_t2_cnt
set firewall family inet filter filter2 term t2_f2 then accept
set firewall family inet filter filter3 term t1_f3 from destination-address 172.30.1.1/32
set firewall family inet filter filter3 term t1_f3 then count f3_t1_cnt
set firewall family inet filter filter3 term t2_f3 from destination-port 5454
set firewall family inet filter filter3 term t2_f3 then count f3_t2_cnt
set firewall family inet filter filter3 term t2_f3 then accept
set interfaces ge-0/1/1 unit 0 family inet address 172.16.1.1/30
set interfaces ge-0/1/1 unit 0 family inet filter input-chain [ filter1 filter2 filter3 ]
set interfaces ge-0/1/1 unit 0 family inet filter output-chain [ filter1 filter2 filter3 ]
```

Configure IPv4 Firewall Filters

Here we configure the firewall filters. Each has different match conditions and count actions. The first two filters have multiple terms with the non-terminating action of **count**, which means matching packets will be passed on to the next filter in the chain, while the third has an action of **accept**. Packets that don't match any of the specified conditions would be dropped.

Step-by-Step Procedure

To configure the firewall filters:

1. Navigate the CLI to the hierarchy level at which you configure IPv4 firewall filters.

```
[edit]
user@host# edit firewall family inet
```

2. Configure the first firewall filter to count TCP packets, or packets with a precedence of 7, before sending them on to the next filter in the chain.

```
[edit firewall family inet]
user@host# set filter filter1 term t1_f1 from protocol tcp
user@host# set filter filter1 term t1_f1 then count f1_t1_cnt
user@host# set filter filter1 term t2_f1 from precedence 7
user@host# set filter filter1 term t2_f1 then count f1_t2_cnt
user@host# set filter filter1 term t2_f1 then accept
```

3. Configure the second firewall filter to count DSCP packets, or packets with a source port of 1020, before sending them on to the next filter in the chain.

```
[edit firewall family inet]
user@host# set filter filter2 term t1_f2 from dscp 0
user@host# set filter filter2 term t1_f2 then count f2_t1_cnt
user@host# set filter filter2 term t2_f2 from source-port 1020
user@host# set filter filter2 term t2_f2 then count f2_t2_cnt
user@host# set filter filter2 term t2_f2 then accept
```

4. Configure the last firewall filter to count and accept packets with a destination address of 172.30.1.1/32, or a destination port of 5454.

```
[edit firewall family inet]
user@host# set filter filter3 term t1_f3 from destination-address 172.30.1.1/32
user@host# set filter filter3 term t1_f3 then count f3_t1_cnt
user@host# set filter filter3 term t2_f3 from destination-port 5454
user@host# set filter filter3 term t2_f3 then count f3_t2_cnt
user@host# set filter filter3 term t2_f3 then accept
```

Apply the Chain of Input Filters

Here we attach the firewall filters to a given interface. The order of execution occurs in the same order as the chain, from left to right.

Step-by-Step Procedure

To assign the interface an IP address:

1. Navigate to the interface we are using for the filters, `ge-0/1/1.0`.

```
[edit]
user@host# edit interfaces ge-0/1/1 unit 0 family inet
```

2. Assign an IPv4 address to the logical interface.

```
[edit interfaces ge-0/1/1 unit 0 family inet]
user@host# set address 172.16.1.1/30
```

3. Apply the filters as a list of input filters.

```
[edit interfaces ge-0/1/1 unit 0 family inet]
user@host# set filter input-chain [ filter1 filter2 filter3 ]
user@host# set filter out-chain [ filter1 filter2 filter3 ]
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filters by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit firewall]
user@host# show
    family inet {
        }
        filter filter1 {
            term t1_f1 {
                from {
                    protocol tcp;
                }
                then count f1_t1_cnt;
                accept;
            }
            term t2_f1 {
                from {
                    precedence 7;
                }
                then count f1_t2_cnt;
                accept;
            }
        }
        filter filter2 {
            term t1_f2 {
                from {
                    dscp 0;
                }
```

```
                then count f2_t1_cnt;
            }
            term t2_f2 {
                from {
                    source-port 1020;
                }
                then count f2_t2_cnt;
            }
        }
        filter filter3 {
            term t1_f3 {
                from {
                    destination-address {
                        172.30.1.1/32;
                    }
                }
                then {
                    count f3_t1_cnt;
                }
            }
            term t2_f3 {
                from {
                    destination-port 5454;
                }
                then {
                    count f3_t2_cnt;
                    accept;
                }
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command.

```
[edit]
user@host# show interfaces
ge-0/1/1 {
    unit 0 {
        family inet {
            filter {
```

```
                input-chain [ filter1 filter2 filter3 ];
            }
            address 172.16.1.1/30;
        }
    }
}
```

3. If you are done configuring the device, commit the configuration.

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

Verification

IN THIS SECTION

- Send Traffic Through the Firewall Filters | 282

Confirm that the configuration works as expected, that is, that the matching traffic is evaluated by each of the filters filter1, filter2, and filter3, and that the expected action (count or accept) has been taken.

Send Traffic Through the Firewall Filters

Purpose

Send traffic from one device to the router you have configured to see whether matching packets are being evaluated by all relevant filters in the chain.

Action

To verify that input packets are evaluated by filter1, filter2, and filter3:

1. From the remote host that is connected to `ge-0/1/1.0`, send a packet with a precedence of 7. The packet should be counted and then evaluated by filter2.
2. From the remote host that is connected to `ge-0/1/1.0`, send a packet with DSCP value of 0. The packet should be counted and then evaluated by filter3.

3. From the remote host that is connected to `ge-0/1/1.0`, send a packet with a destination address of 172.30.1.1/32 and a destination port number of 5454. The packet should be counted and then accepted.

4. To display counter information for the filters you configured, enter the `show firewall filter` *`filter-name`* operational mode command. The command output displays the number of bytes and packets that match filter terms associated with the counters.

RELATED DOCUMENTATION

output-chain

input-chain

Understanding Policy Subroutines in Routing Policy Match Conditions

IN THIS SECTION

- Configuring Subroutines | 283

You can use a routing policy called from another routing policy as a match condition. This process makes the called policy a *subroutine*.

In some ways, the Junos OS policy framework is similar to a programming language. This similarity includes the concept of nesting policies into a policy subroutine. A subroutine in a software program is a section of code that you reference on a regular basis. A policy subroutine works in the same fashion—you reference an existing policy as a match criterion in another policy. The routing device first evaluates the subroutine and then evaluates the main policy. The evaluation of the subroutine returns a true or false Boolean result to the main policy. Because you are referencing the subroutine as a match criterion, a true result means that the main policy has a match and can perform any configured actions. A false result from the subroutine, however, means that the main policy does not have a match.

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 in the `from` or `to` statement of another routing policy.

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.

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.

In particular, you must consider what happens if a match does not occur with routes specified in a subroutine and if the default policy action that is taken is the action that you expect and want.

For example, imagine that you are a network administrator at an Internet service provider (ISP) that provides service to Customer A. You have configured several routing policies for the different classes of neighbors that Customer A presents on various links. To save time maintaining the routing policies for Customer A, you have configured a subroutine that identifies their routes and various routing policies that call the subroutine, as shown below:

```
[edit]
policy-options {
    policy-statement customer-a-subroutine {
        from {
            route-filter 10.1/16 exact;
            route-filter 10.5/16 exact;
            route-filter 192.168.10/24 exact;
        }
        then accept;
    }
}
policy-options {
    policy-statement send-customer-a-default {
        from {
            policy customer-a-subroutine;
        }
```

```
        then {
            set metric 500;
            accept;
        }
    }
}
policy-options {
    policy-statement send-customer-a-primary {
        from {
            policy customer-a-subroutine;
        }
        then {
            set metric 100;
            accept;
        }
    }
}
policy-options {
    policy-statement send-customer-a-secondary {
        from {
            policy customer-a-subroutine;
        }
        then {
            set metric 200;
            accept;
        }
    }
}
protocols {
    bgp {
        group customer-a {
            export send-customer-a-default;
            neighbor 10.1.1.1;
            neighbor 10.1.2.1;
            neighbor 10.1.3.1 {
                export send-customer-a-primary;
            }
            neighbor 10.1.4.1 {
                export send-customer-a-secondary;
            }
        }
```

```
    }
}
```

The following results occur with this configuration:

- The group-level `export` statement resets the metric to 500 when advertising all BGP routes to neighbors 10.1.1.1 and 10.1.2.1 rather than just the routes that match the subroutine route filters.
- The neighbor-level `export` statements reset the metric to 100 and 200 when advertising all BGP routes to neighbors 10.1.3.1 and 10.1.4.1, respectively, rather than just the BGP routes that match the subroutine route filters.

These unexpected results occur because the subroutine policy does not specify a termination action for routes that do not match the route filter and therefore, the default BGP export policy of accepting all BGP routes is taken.

If the statements included in this particular subroutine had been contained within the calling policies themselves, only the desired routes would have their metrics reset.

This example illustrates the differences between routing policies and subroutines and the importance of the termination action in a subroutine. Here, the default BGP export policy action for the subroutine was not carefully considered. A solution to this particular example is to add one more term to the subroutine that rejects all other routes that do not match the route filters:

```
[edit]
policy-options {
    policy-statement customer-a-subroutine {
        term accept-exact {
            from {
                route-filter 10.1/16 exact;
                route-filter 10.5/16 exact;
                route-filter 192.168.10/24 exact;
            }
            then accept;
        }
        term reject-others {
            then reject;
        }
    }
}
```

Termination action strategies for subroutines in general include the following:

- Depend upon the default policy action to handle all other routes.

- Add a term that accepts all other routes.
- Add a term that rejects all other routes.

The option that you choose depends upon what you want to achieve with your subroutine. Plan your subroutines carefully.

RELATED DOCUMENTATION

How a Routing Policy Subroutine Is Evaluated | 287

Example: Configuring a Policy Subroutine | 289

How a Routing Policy Subroutine Is Evaluated

Figure 17 on page 288 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" on page 287). If the route matches all match conditions specified before the subroutine, the route is evaluated against the subroutine. If the route matches the match conditions in any of the subroutine terms, two levels of evaluation occur in the following order:

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

      If the subroutine does not specify the `accept`, `reject` or `next-policy` action, it uses the `accept` or `reject` action specified by the default policy, and the values of TRUE or FALSE are returned to the calling policy as described in the previous paragraph.

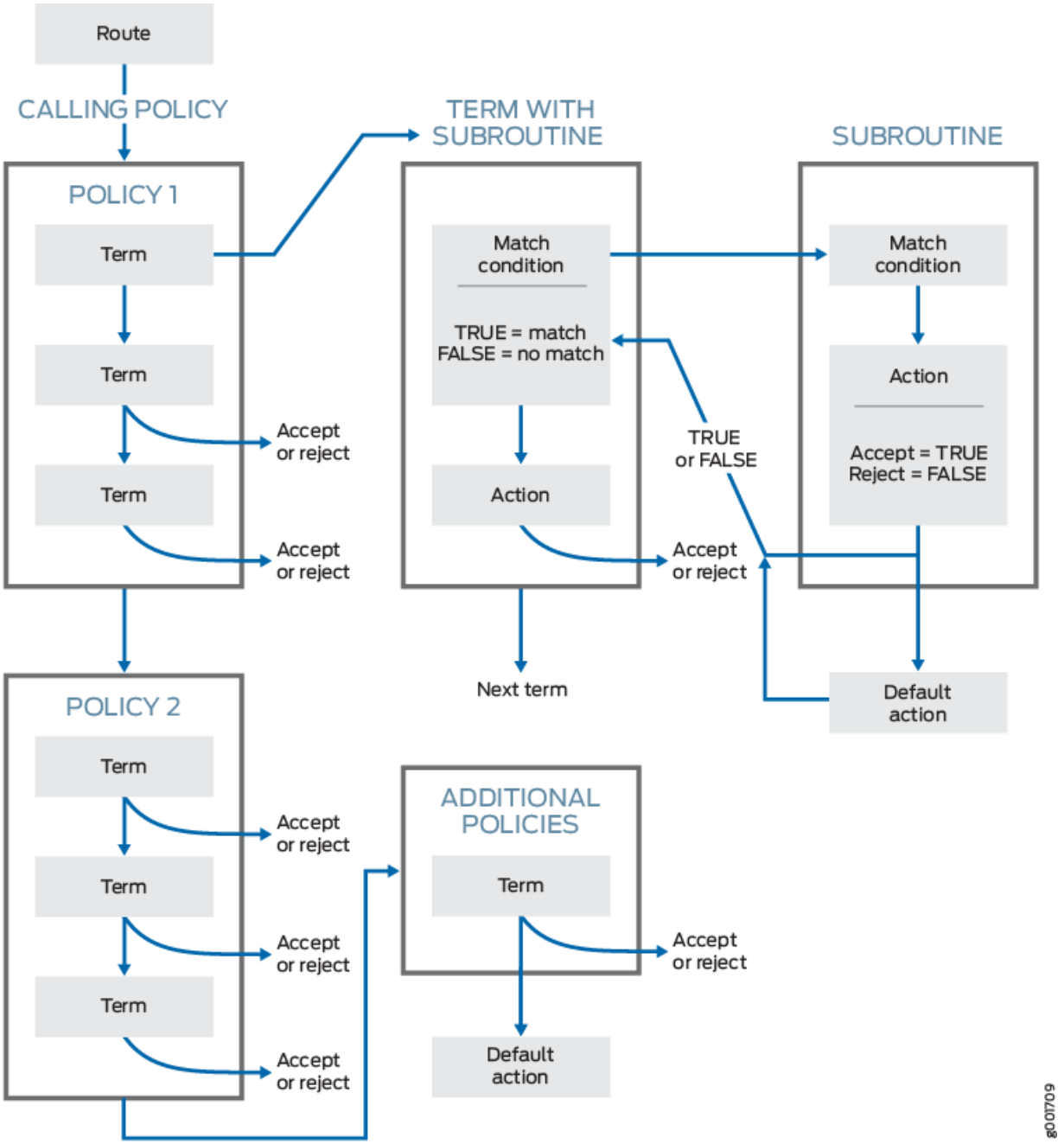
   b. The calling policy's subroutine match condition is evaluated. During this part of the evaluation, TRUE equals a match and FALSE equals no match. If the subroutine returns TRUE to the calling policy, then the evaluation of the calling policy continues. If the subroutine returns FALSE to the calling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

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.

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 17: Routing Policy Subroutine Evaluation



RELATED DOCUMENTATION

Default Routing Policies | 40

Understanding Policy Subroutines in Routing Policy Match Conditions | 283

Understanding How a Routing Policy Chain Is Evaluated | 256

Example: Configuring a Policy Subroutine | 289

Example: Configuring a Policy Subroutine

IN THIS SECTION

- Requirements | 289
- Overview | 289
- Configuration | 292
- Verification | 299

This example demonstrates the use of a policy subroutine in a routing policy match condition.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 291

On Device R1, a policy called `main` is configured.

```
user@R1# show policy-options
policy-statement main {
    term subroutine-as-a-match {
```

```
        from policy subroutine;
        then accept;
    }
    term nothing-else {
        then reject;
    }
}
```

This main policy calls a subroutine called `subroutine`.

```
user@R1# show policy-options
policy-statement subroutine {
    term get-routes {
        from protocol static;
        then accept;
    }
    term nothing-else {
        then reject;
    }
}
```

The router evaluates the logic of `main` in a defined manner. The match criterion of `from policy subroutine` allows the routing device to locate the subroutine. All terms of the subroutine are evaluated, in order, following the normal policy processing rules. In this example, all static routes in the routing table match the subroutine with an action of accept. This returns a true result to the original, or calling, policy which informs the device that a positive match has occurred. The actions in the calling policy are executed and the route is accepted. All other routes in the routing table do not match the subroutine and return a false result to the calling policy. The device evaluates the second term of `main` and rejects the routes.

The actions in the subroutine do not actually accept or reject a specific route. The subroutine actions are only translated into a true or a false result. Actions that modify a route's attributes, however, are applied to the route regardless of the outcome of the subroutine.

Device R1 in AS 64510 has multiple customer routes, some of which are static routes configured locally, and some of which are received from Device R2 and Device R3 through internal BGP (IBGP). AS 64510 is connected to Device R4 in AS 64511. The policy `main` is applied as an export policy in Device R1's BGP peering session with Device R4. This causes Device R1 to send only its own static routes to Device R4. Because of the policy `main`, Device R1 does not send the routes received from its internal peers, Device R2 and Device R3.

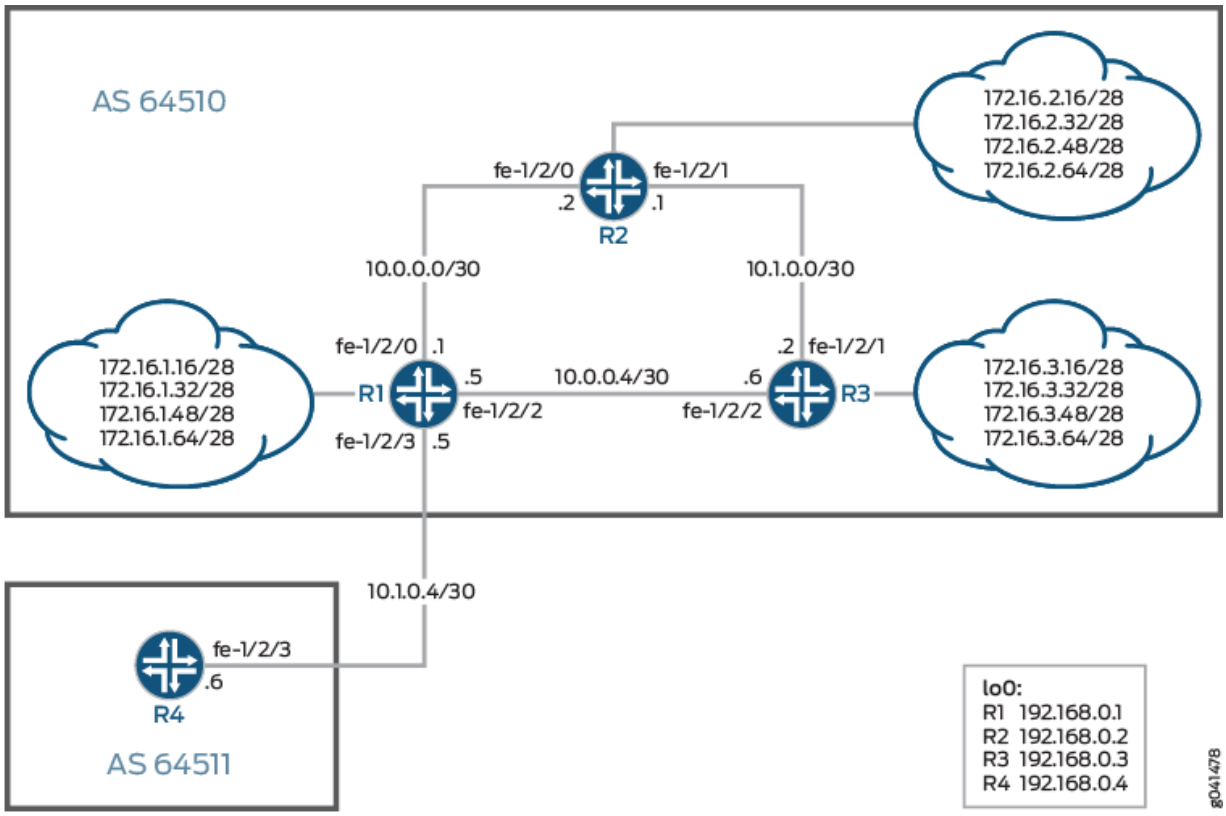
When you are working with policy subroutines, it is important to remember that the default EBGP export policy is to advertise all learned BGP routes to all EBGP peers. This default policy is in effect in the main policy and also in the subroutine. Therefore, as shown in this example, if you do not want the

default EBGP export policy to take effect, you must configure a `then reject` terminating action as the final term in both the main policy and in the policy subroutine. This example demonstrates what happens when the final `then reject` term is missing either from the main policy or from the policy subroutine.

Topology

Figure 18 on page 291 shows the sample network.

Figure 18: BGP Topology for Policy Subroutine



"CLI Quick Configuration" on page 292 shows the configuration for all of the devices in Figure 18 on page 291.

The section "No Link Title" on page 294 describes the steps on Device R1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 292
- Procedure | 294

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 description to_R2
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/2 unit 0 description to_R3
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.5/30
set interfaces fe-1/2/3 unit 0 description to_R4
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.5/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.1
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group to_64511 type external
set protocols bgp group to_64511 export main
set protocols bgp group to_64511 neighbor 10.1.0.6 peer-as 64511
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/2.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement main term subroutine-as-a-match from policy subroutine
set policy-options policy-statement main term subroutine-as-a-match then accept
set policy-options policy-statement main term nothing-else then reject
set policy-options policy-statement subroutine term get-routes from protocol static
set policy-options policy-statement subroutine term get-routes then accept
set policy-options policy-statement subroutine term nothing-else then reject
set routing-options static route 172.16.1.16/28 discard
```

```
set routing-options static route 172.16.1.32/28 discard
set routing-options static route 172.16.1.48/28 discard
set routing-options static route 172.16.1.64/28 discard
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device R2

```
set interfaces fe-1/2/0 unit 0 description to_R1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 description to_R3
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int neighbor 192.168.0.1 export send-static
set protocols bgp group int neighbor 192.168.0.3
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface fe-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 172.16.2.16/28 discard
set routing-options static route 172.16.2.32/28 discard
set routing-options static route 172.16.2.48/28 discard
set routing-options static route 172.16.2.64/28 discard
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64510
```

Device R3

```
set interfaces fe-1/2/1 unit 0 description to_R2
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces fe-1/2/2 unit 0 description to_R1
set interfaces fe-1/2/2 unit 0 family inet address 10.0.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int neighbor 192.168.0.1 export send-static
set protocols bgp group int neighbor 192.168.0.2
set protocols ospf area 0.0.0.0 interface fe-1/2/2.6
```

```
set protocols ospf area 0.0.0.0 interface fe-1/2/0.4
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static from protocol static
set policy-options policy-statement send-static then accept
set routing-options static route 172.16.3.16/28 discard
set routing-options static route 172.16.3.32/28 discard
set routing-options static route 172.16.3.48/28 discard
set routing-options static route 172.16.3.64/28 discard
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64510
```

Device R4

```
set interfaces fe-1/2/3 unit 0 description to_R1
set interfaces fe-1/2/3 unit 0 family inet address 10.1.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.1.0.5
set routing-options autonomous-system 64511
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 0 description to_R2
user@R1# set fe-1/2/0 unit 0 family inet address 10.0.0.1/30
user@R1# set fe-1/2/2 unit 0 description to_R3
user@R1# set fe-1/2/2 unit 0 family inet address 10.0.0.5/30
user@R1# set fe-1/2/3 unit 0 description to_R4
```

```
user@R1# set fe-1/2/3 unit 0 family inet address 10.1.0.5/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure the internal BGP (IBGP) connections to Device R2 and Device R3.

```
[edit protocols bgp group int]
user@R1# set type internal
user@R1# set local-address 192.168.0.1
user@R1# set neighbor 192.168.0.2
user@R1# set neighbor 192.168.0.3
```

3. Configure the EBGP connection to Device R4.

```
[edit protocols bgp group to_64511]
user@R1# set type external
user@R1# set export main
user@R1# set neighbor 10.1.0.6 peer-as 64511
```

4. Configure OSPF connections to Device R2 and Device R3.

```
[edit protocols ospf area 0.0.0.0]
user@R1# set interface fe-1/2/0.0
user@R1# set interface fe-1/2/2.0
user@R1# set interface lo0.0 passive
```

5. Configure the policy `main`.

```
[edit policy-options policy-statement main term subroutine-as-a-match]
user@R1# set from policy subroutine
user@R1# set then accept
[edit policy-options policy-statement main term nothing-else]
user@R1# set then reject
```

6. Configure the policy `subroutine`.

```
[edit policy-options policy-statement subroutine term get-routes]
user@R1# set from protocol static
user@R1# set then accept
```

```
[edit policy-options policy-statement subroutine term nothing-else]
user@R1# set then reject
```

7. Configure the static route to the 172.16.5.0/24 network.

```
[edit routing-options static]
user@R1# set route 172.16.1.16/28 discard
user@R1# set route 172.16.1.32/28 discard
user@R1# set route 172.16.1.48/28 discard
user@R1# set route 172.16.1.64/28 discard
```

8. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R1# set router-id 192.168.0.1
user@R1# set autonomous-system 64510
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
fe-1/2/0 {
    unit 0 {
        description to_R2;
        family inet {
            address 10.0.0.1/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        description to_R3;
        family inet {
            address 10.0.0.5/30;
        }
    }
```

```
}
fe-1/2/3 {
    unit 0 {
        description to_R4;
        family inet {
            address 10.1.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.1;
        neighbor 192.168.0.2;
        neighbor 192.168.0.3;
    }
    group to_64511 {
        type external;
        export main;
        neighbor 10.1.0.6 {
            peer-as 64511;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/0.0;
        interface fe-1/2/2.0;
        interface lo0.0 {
            passive;
        }
```

```
    }
}
```

```
user@R1# show policy-options
policy-statement main {
    term subroutine-as-a-match {
        from policy subroutine;
        then accept;
    }
    term nothing-else {
        then reject;
    }
}
policy-statement subroutine {
    term get-routes {
        from protocol static;
        then accept;
    }
    term nothing-else {
        then reject;
    }
}
```

```
user@R1# show routing-options
static {
    route 172.6.1.16/28 discard;
    route 172.6.1.32/28 discard;
    route 172.6.1.48/28 discard;
    route 172.6.1.64/28 discard;
}
router-id 192.168.0.1;
autonomous-system 64510;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes on Device R1 | 299
- Verifying the Route Advertisement to Device R4 | 300
- Experimenting with the Default BGP Export Policy | 300

Confirm that the configuration is working properly.

Verifying the Routes on Device R1

Purpose

On Device R1, check the static routes in the routing table.

Action

```
user@R1> show route protocol static

inet.0: 23 destinations, 23 routes (23 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.1.16/28     *[Static/5] 1d 02:02:13
                      Discard
172.16.1.32/28     *[Static/5] 1d 02:02:13
                      Discard
172.16.1.48/28     *[Static/5] 1d 02:02:13
                      Discard
172.16.1.64/28     *[Static/5] 1d 02:02:13
                      Discard
```

Meaning

Device R1 has four static routes.

Verifying the Route Advertisement to Device R4

Purpose

On Device R1, make sure that the static routes are advertised to Device R4.

Action

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 23 destinations, 23 routes (23 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
* 172.16.1.64/28          Self                                    I
```

Meaning

As expected, Device R1 only advertises its static routes to Device R4.

Experimenting with the Default BGP Export Policy

Purpose

See what can happen when you remove the final `then reject` term from the policy `main` or the policy `subroutine`.

Action

1. On Device R1, deactivate the final term in the policy `main`.

```
[edit  policy-options policy-statement main]
user@R1# deactivate term nothing-else
user@R1# commit
```

2. On Device R1, check to see which routes are advertised to Device R4.

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 23 destinations, 23 routes (23 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
* 172.16.1.64/28          Self                                    I
* 172.16.2.16/28          Self                                    I
* 172.16.2.32/28          Self                                    I
* 172.16.2.48/28          Self                                    I
* 172.16.2.64/28          Self                                    I
* 172.16.3.16/28          Self                                    I
* 172.16.3.32/28          Self                                    I
* 172.16.3.48/28          Self                                    I
* 172.16.3.64/28          Self                                    I
```

Now, all the BGP routes from Device R1 are sent to Device R4. This is because after the processing is returned to policy `main`, the default BGP export policy takes effect.

3. On Device R1, reactivate the final term in the policy `main`, and deactivate the final term in the policy `subroutine`.

```
[edit  policy-options policy-statement main]
user@R1# activate term nothing-else

[edit  policy-options policy-statement subroutine]
user@R1# deactivate term nothing-else
user@R1# commit
```

4. On Device R1, check to see which routes are advertised to Device R4.

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 23 destinations, 23 routes (23 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
```

```
* 172.16.1.64/28          Self                          I
* 172.16.2.16/28          Self                          I
* 172.16.2.32/28          Self                          I
* 172.16.2.48/28          Self                          I
* 172.16.2.64/28          Self                          I
* 172.16.3.16/28          Self                          I
* 172.16.3.32/28          Self                          I
* 172.16.3.48/28          Self                          I
* 172.16.3.64/28          Self                          I
```

Now, all the BGP routes from Device R1 are sent to Device R4. This is because before the processing is returned to policy `main`, the default BGP export policy takes effect in the policy `subroutine`.

Meaning

To prevent the default BGP export policy from taking effect, you must include a final `then reject` term in the main policy and in all referenced subroutines.

RELATED DOCUMENTATION

Understanding Policy Subroutines in Routing Policy Match Conditions | 283

How a Routing Policy Subroutine Is Evaluated | 287

CHAPTER 4

Configuring Route Filters and Prefix Lists as Match Conditions

IN THIS CHAPTER

- Understanding Route Filters for Use in Routing Policy Match Conditions | 303
- Understanding Route Filter and Source Address Filter Lists for Use in Routing Policy Match Conditions | 327
- Understanding Load Balancing Using Source or Destination IP Only | 327
- Configuring Load Balancing Using Source or Destination IP Only | 328
- Walkup for Route Filters Overview | 330
- Configuring Walkup for Route Filters to Improve Operational Efficiency | 334
- Example: Configuring Route Filter Lists | 340
- Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency | 346
- Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency | 354
- Example: Configuring a Route Filter Policy to Specify Priority for Prefixes Learned Through OSPF | 362
- Example: Configuring the MED Using Route Filters | 368
- Example: Configuring Layer 3 VPN Protocol Family Qualifiers for Route Filters | 388
- Understanding Prefix Lists for Use in Routing Policy Match Conditions | 392
- Example: Configuring Routing Policy Prefix Lists | 396
- Example: Configuring the Priority for Route Prefixes in the RPD Infrastructure | 412
- Configuring Priority for Route Prefixes in RPD Infrastructure | 426

Understanding Route Filters for Use in Routing Policy Match Conditions

IN THIS SECTION

- Radix Trees | 304

- Configuring Route Filters | **306**
- How Route Filters Are Evaluated in Routing Policy Match Conditions | **314**
- Route Filter Examples | **317**

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

NOTE: Because the configuration of route filters includes setting up prefixes and prefix lengths, before proceeding with the configuration you should have a thorough understanding of IP addressing, including supernetting, and how route filters are evaluated (explained here: "How Route Filters Are Evaluated in Routing Policy Match Conditions" on page 314).

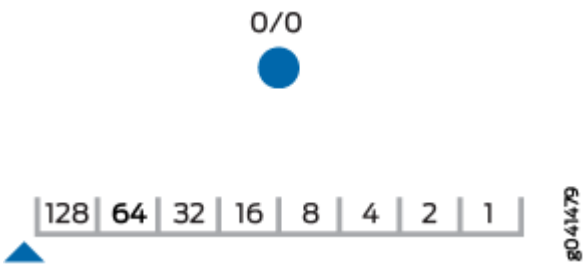
This section discusses the following topics:

Radix Trees

To understand the operation of a route filter, you need to be familiar with a device used for binary number matching known as a radix tree (sometimes called a patricia trie or radix trie). A radix tree uses binary lookups to identify IP addresses (routes). Remember that an IP address is a 32-bit number represented in a dotted decimal format for easy comprehension by humans. These 8-bit groupings can each have a value between 0 and 255. A radix tree can be a graphical representation of these binary numbers.

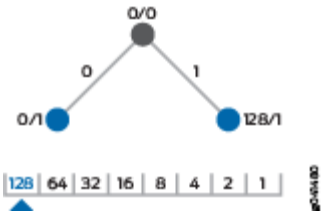
In Figure 19 on page 304, the radix tree starts with no configured value (starts at 0) and is at the leftmost position of the binary IP address. This is shown as 0/0, which is often referred to as the default route.

Figure 19: Beginning of a Radix Tree



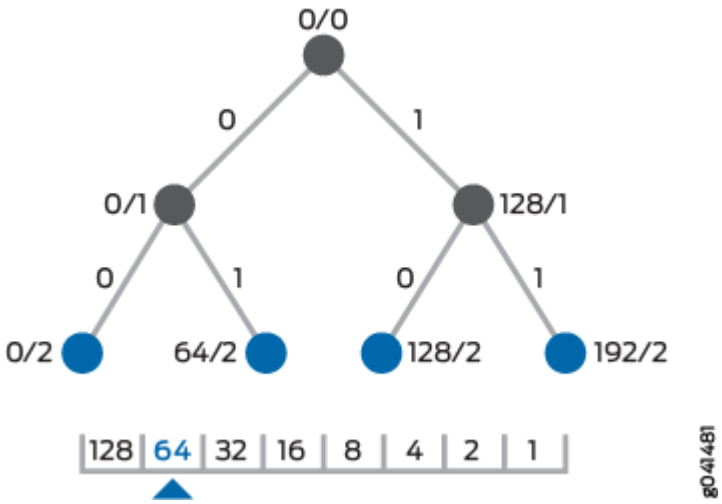
Because this is binary, each bit can have only one of two possible values—a 0 or a 1. Moving down the left branch represents a value of 0, while moving to the right represents a value of 1. The first step is shown in Figure 20 on page 305. At the first position, the first octet of the IP address has a value of 00000000 or 10000000—a 0 or 128, respectively. This is represented in Figure 20 on page 305 by the values 0/1 and 128/1.

Figure 20: First Step of a Radix Tree



The second step is shown in Figure 21 on page 305. This second level of the tree has four possible binary values for the first octet: 00000000, 01000000, 10000000, and 11000000. These decimal values of 0, 64, 128, and 192 are represented by the IP addresses of 0/2, 64/2, 128/2, and 192/2 on the radix tree.

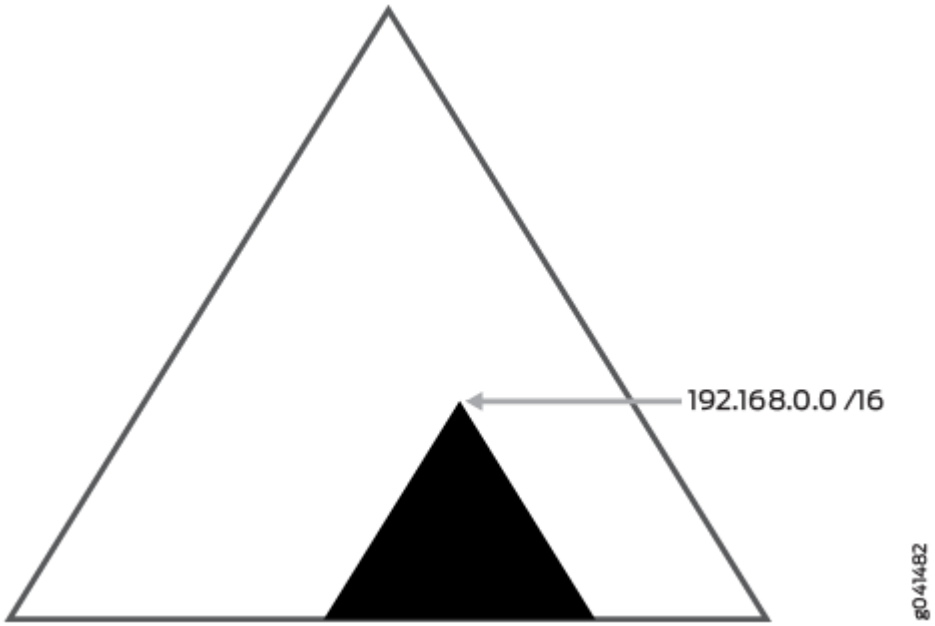
Figure 21: Second Step of a Radix Tree



This step-by-step process continues for 33 total levels to represent every possible IP address.

The radix tree structure is helpful when locating a group of routes that all share the same most significant bits. Figure 22 on page 306 shows the point in the radix tree that represents the 192.168.0.0/16 network. All of the routes that are more specific than 192.168.0.0/16 are shown in the highlighted section.

Figure 22: Locating a Group of Routes



Configuring Route Filters

NOTE: The topic, Configuring Route Filters, describes default Junos OS behavior. The walkup feature, which is not covered in this topic, alters the evaluation results discussed in this topic by allowing the router to consider shorter match conditions configured within the same term. See "Walkup for Route Filters Overview" on page 330 for details.

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

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

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

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

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

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

source-prefix address is the IPv4 or IPv6 address prefix specified as *prefix*/*prefix-length*. If you omit *prefix-length* for an IPv4 prefix, the default is /32*prefix-length*. If you omit *prefix-length* for an IPv6 prefix, the default is /128. Prefixes specified in a from statement must be either all IPv4 addresses or all IPv6 addresses.

match-type is the type of match to apply to the source or destination prefix. It can be one of the match types listed in Table 14 on page 308. For examples of the match types and the results when presented with various routes, see Table 15 on page 313.

actions are the actions to take if a route address matches the criteria specified for a destination match prefix (specified as part of a route-filter option) or for a source match prefix (specified as part of a destination-address-filter option). The actions can consist of one or more of the actions described in "Actions in Routing Policy Terms" on page 75.

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

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

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

For more examples of these route filter match types, see "Route Filter Examples" on page 317.

Table 14: Route Filter Match Types for a Prefix List

Match Type	Match Criteria
address-mask *netmask-value*	All of the following are true: • The bit-wise logical AND of the *netmask-value* pattern and the incoming IPv4 or IPv6 route address and the bit-wise logical AND of the *netmask-value* pattern and the *destination-prefix* address are the same. The bits set in the *netmask-value* pattern do not need to be contiguous. • The *prefix-length* component of the incoming IPv4 or IPv6 route address and the *prefix-length* component of the *destination-prefix* address are the same. **NOTE**: The address-mask routing policy match type is valid only for matching an incoming IPv4 (family inet) or IPv6 (family inet6) route address to a list of destination match prefixes specified in a route-filter statement. The address-mask routing policy match type enables you to match an incoming IPv4 or IPv6 route address on a configured netmask address in addition to the length of a configured destination match prefix. The length of the route address must match exactly with the length of the configured destination match prefix, as the address-mask match type does not support prefix length variations for a range of prefix lengths. When the longest-match lookup is performed on a route filter, the lookup evaluates an address-mask match type differently from other routing policy match types. The lookup does not consider the length of the destination match prefix. Instead, the lookup considers the number of contiguous high-order bits set in the netmask value. For more information about this route filter match type, see "How an Address Mask Match Type Is Evaluated" on page 316. For example configurations showing route filters that contain the address-mask match type, see the following topics: • "Accepting Incoming IPv4 Routes by Applying an Address Mask to the Route Address and the Destination Match Prefix" on page 323. • "Accepting Incoming IPv4 Routes with Similar Patterns But Different Prefix Lengths" on page 324. • "Evaluation of an Address Mask Match Type with Longest-Match Lookup" on page 325.

Table 14: Route Filter Match Types for a Prefix List *(Continued)*

Match Type	Match Criteria
`exact`	All of the following are true: • The route address shares the same most-significant bits as the match prefix (*destination-prefix* or *source-prefix*). The number of significant bits is described by the *prefix-length* component of the match prefix. • The *prefix-length* component of the match prefix is equal to the route's prefix length.
`longer`	All of the following are true: • The route address shares the same most-significant bits as the match prefix (*destination-prefix* or *source-prefix*). The number of significant bits is described by the *prefix-length* component of the match prefix. • The route's prefix length is greater than the *prefix-length* component of the match prefix.
`orlonger`	All of the following are true: • The route address shares the same most-significant bits as the match prefix (*destination-prefix* or the *source-prefix*). The number of significant bits is described by the *prefix-length* component of the match prefix. • The route's prefix length is equal to or greater than the *prefix-length* component of the configured match prefix.
`prefix-length-range` *prefix-length2-prefix-length3*	All of the following are true: • The route address shares the same most-significant bits as the match prefix (*destination-prefix* or *source-prefix*). The number of significant bits is described by the *prefix-length* component of the match prefix. • The route's prefix length falls between *prefix-length2* and *prefix-length3*, inclusive.

Table 14: Route Filter Match Types for a Prefix List *(Continued)*

Match Type	Match Criteria
through {*destination-prefix2* | *source-prefix2*}	All of the following are true: • The route address shares the same most-significant bits as the first match prefix (*destination-prefix* or *source-prefix*). The number of significant bits is described by the *prefix-length* component of the first match prefix. • The route address shares the same most-significant bits as the second match prefix (*destination-prefix2* or *source-prefix2*). The number of significant bits is described by the *prefix-length* component of the second match prefix. • The route's prefix length is less than or equal to the *prefix-length* component of the second match prefix. You do not use the through match type in most routing policy configurations. For an example, see "Rejecting Routes from Specific Hosts" on page 319.
upto *prefix-length2*	All of the following are true: • The route address shares the same most-significant bits as the match prefix (*destination-prefix* or *source-prefix*). The number of significant bits is described by the *prefix-length* component of the match prefix. • The route's prefix length falls between the *prefix-length* component of the first match prefix and *prefix-length2*.

Figure 23 on page 311 shows the detailed radix tree for the route 192.168.0.0/16.

Figure 23: Portion of the Radix Tree

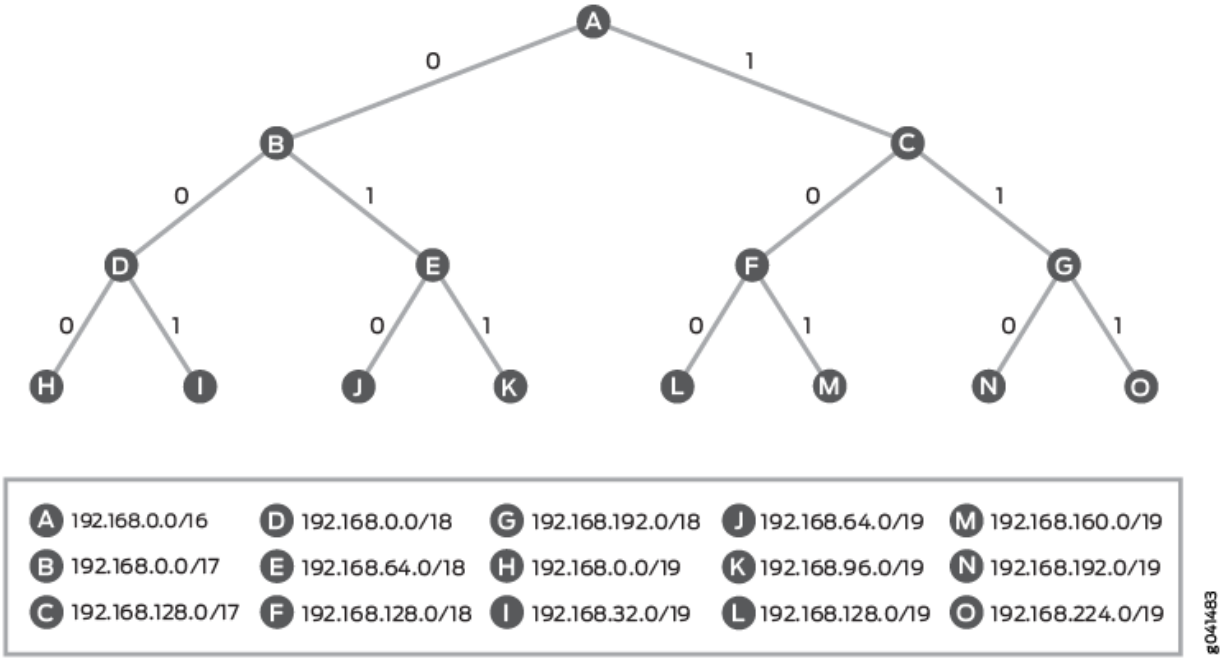


Figure 24 on page 312 and Table 15 on page 313 demonstrate the operation of the various route filter match types.

Figure 24: Route Filter Match Types

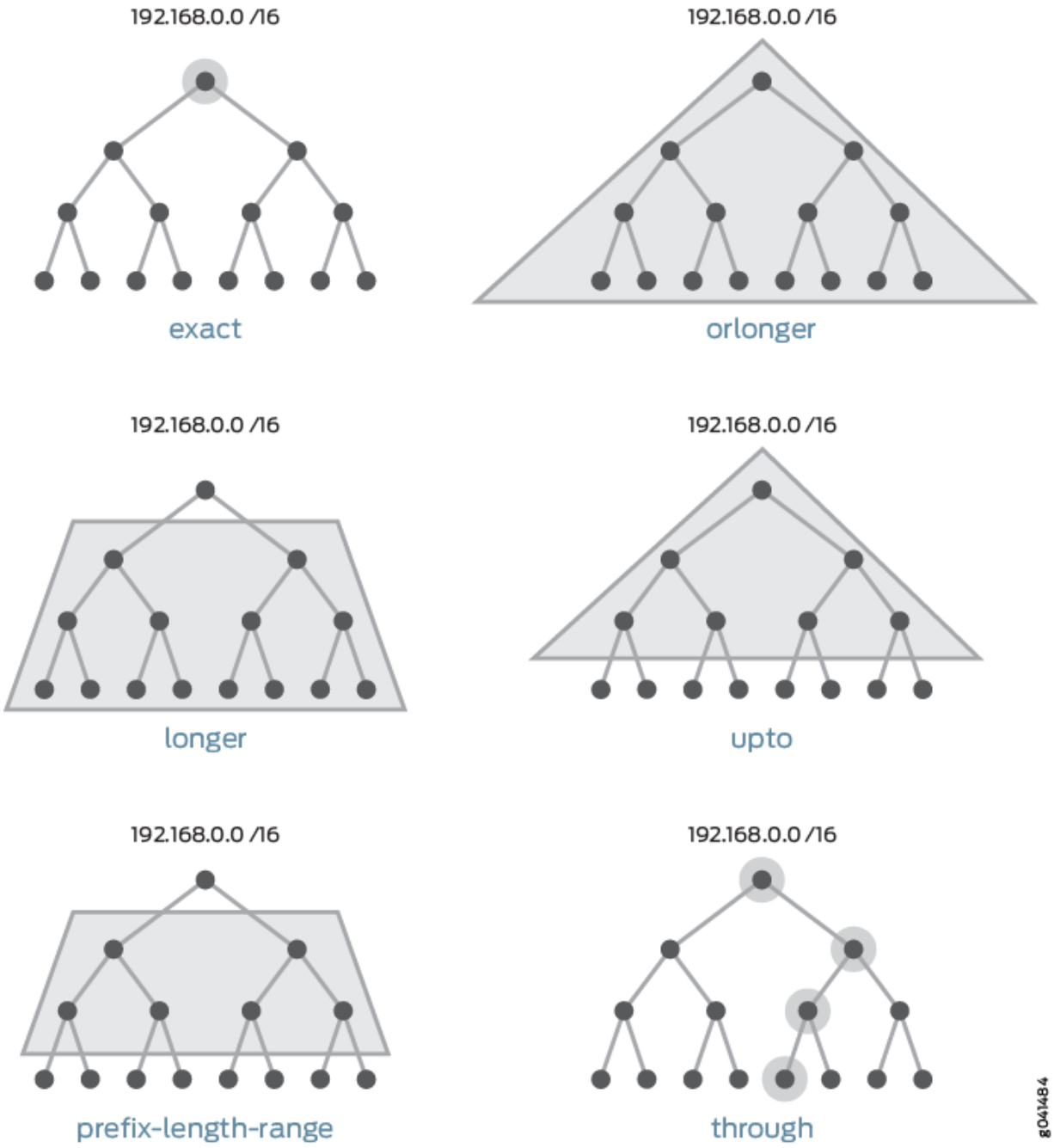


Table 15: Match Type Examples

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 prefix-length-range/18 – /20	192.168/16 through192.168.16/20	192.168/19 address-mask255.255.0.0
10.0.0.0/8	–	–	–	–	–	–	–
192.168.0.0/16	Match	–	Match	Match	–	Match	–
192.168.0.0/17	–	Match	Match	Match	–	Match	–
192.168.0.0/18	–	Match	Match	Match	Match	Match	–
192.168.0.0/19	–	Match	Match	Match	Match	Match	Match
192.168.4.0/24	–	Match	Match	Match	–	–	–
192.168.5.4/30	–	Match	Match	–	–	–	–
192.168.12.4/30	–	Match	Match	–	–	–	–
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	–	–

Table 15: Match Type Examples *(Continued)*

Prefix	192.168/16 exact	192.168/16 longer	192.168/16 orlonger	192.168/16 upto /24	192.168/16 prefix-length-range/18 – /20	192.168/16 through192.168.16/20	192.168/19 address-mask255.255.0.0
192.168.224.0/19	–	Match	Match	Match	Match	–	Match
10.169.1.0/24	–	–	–	–	–	–	–
10.170.0.0/16	–	–	–	–	–	–	–

How Route Filters Are Evaluated in Routing Policy Match Conditions

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

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

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

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

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

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

   a. The route filter stops evaluating other prefixes, even if the match type fails.

   b. The software examines the match type and action associated with that prefix.

NOTE: When a route source address is evaluated against a match criteria that uses the `address-mask` match type, both steps of the evaluation include the configured netmask value. For more information, see "How an Address Mask Match Type Is Evaluated" on page 316.

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" on page 40.

NOTE: If you specify multiple prefixes in the route filter, only one prefix needs to match for a match to occur. The route filter 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 filter:

```
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.

NOTE: The walkup feature allows terms with multiple route filters to “walk-up” the evaluation process to include less-specific routes as well as the longest match. In other words, enabling walkup changes the default behavior from “if one fails, then the term fails” to “if one matches,

then the term matches." For more information about the *walkup* feature, see "Walkup for Route Filters Overview" on page 330.

How Prefix Order Affects Route Filter 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 filter 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.

NOTE: The walkup feature allows terms with multiple route filters to "walk-up" the evaluation process to include less-specific routes as well as the longest match. In other words, enabling walkup changes the default behavior from "if one fails, then the term fails" to "if one matches, then the term matches." For more information about the *walkup* feature, see "Walkup for Route Filters Overview" on page 330.

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;
```

How an Address Mask Match Type Is Evaluated

The `address-mask` routing policy match type enables you to match incoming IPv4 or IPv6 route addresses on a configured netmask value in addition to the length of a configured destination match prefix. During route filter evaluation, an `address-mask` match type is processed differently from other routing policy match types, taking into consideration the configured netmask value:

- When a longest-match lookup evaluates an `address-mask` routing policy match type, the *`prefix-length`* component of the configured match prefix is not considered. Instead, the lookup considers the number of contiguous high-order bits set in the configured netmask value.
- When an incoming IPv4 or IPv6 route address is evaluated against a route filter match criteria that uses the `address-mask` routing policy match type, the match succeeds if the following values are identical:

- The bit-wise logical AND of the configured netmask value and the incoming IPv4 or IPv6 route address
- The bit-wise logical AND of the configured netmask value and the configured destination match prefix

For an example configuration of a route filter that contains two `address-mask` match types, see "Evaluation of an Address Mask Match Type with Longest-Match Lookup" on page 325.

Common Configuration Problem with the Longest-Match Lookup

A common problem when defining a route filter 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 filter:

```
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" on page 40.) The shorter prefix 192.168.0.0/16 orlonger that you wanted to match is inadvertently ignored.

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

Another solution is to enable the *walkup* feature. See "Walkup for Route Filters Overview" on page 330 for details.

Route Filter 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" on page 40.

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

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;
    }
}
```

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;
        }
    }
}
```

Rejecting Routes with Mask Length Between 26 and 29

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

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

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
```

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;
        }
    }
}
```

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 172.16.233.0/3 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
            }
        }
    }
}
```

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.

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;
    }
}
```

Rejecting PIM Traffic

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

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

The following routing policy qualifiers apply to PIM:

- `interface`—Interface over which a join is received
- `neighbor`—Source from which a join originates
- `route-filter`—Group address
- `source-address-filter`—Source address for which to reject a join

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

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

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

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

The route filter in routing policy term `term_1` matches the following incoming IPv4 route addresses:

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

The bit-wise logical AND of the netmask value and the candidate route address must match the bit-wise logical AND of the netmask value and the match prefix address. That is, where the netmask bit pattern

255.255.241.0 contains a set bit, the incoming IPv4 route address being evaluated must match the value of the corresponding bit in the destination prefix address 10.1.0.0/24.

- The first two bytes of the netmask value are binary 1111 1111 1111 1111, which means that a candidate route address will fail the match if the first two bytes are not 10.1.
- The third byte of the netmask value is binary 1111 0001, which means that a candidate route address will fail the match if the third byte is greater than 15 (decimal), an odd number, or both.
- The prefix length of the match prefix address is 24 (decimal), which means that a candidate route address will fail the match if its prefix length is not exactly 24.

As an example, suppose that the candidate route address being tested in the policy is 10.1.8.0/24 (binary 0000 1010 0000 0001 0000 1000).

- When the netmask value is applied to this candidate route address, the result is binary 0000 1010 0000 0001 0000 0000.
- When the netmask value is applied to the configured destination prefix address, the result is also binary 0000 1010 0000 0001 0000 0000.
- Because the results of both AND operations are the same, the match continues to the second match criteria.
- Because the prefix lengths of the candidate address and the configured destination prefix address are the same (24 bits), the match succeeds.

As another example, suppose that the candidate route address being tested in the policy is 10.1.3.0/24 (binary 0000 1010 0000 0001 0000 0011).

- When the netmask value is applied to this candidate route address, the result is binary 0000 1010 0000 0001 0000 0001.
- However, when the netmask value is applied to the configured destination prefix address, the result is binary 0000 1010 0000 0001 0000 0000.
- Because the results of the two AND operations are different (in the third byte), the match fails.

Accepting Incoming IPv4 Routes with Similar Patterns But Different Prefix Lengths

Accept incoming IPv4 route addresses of the form 10.*.1/24 or 10.*.1.*/32:

```
[edit]
policy-options {
    policy-statement from_customer_b {
        term term_2 {
```

```
            from {
                route-filter 10.0.1.0/24 address-mask 255.0.255.0;
                route-filter 10.0.1.0/32 address-mask 255.0.255.0;
            }
            then {
                ...
                reject;
            }
        }
    }
}
```

The route filter match criteria `10.0.1.0/24 address-mask 255.0.255.0` matches an incoming IPv4 route address of the form 10.*.1/24. The route's prefix length must be exactly 24 bits long, and any value is acceptable in the second byte.

The route filter match criteria `10.0.1.0/32 address-mask 255.0.255.0` matches an incoming IPv4 route address of the form 10.*.1.*/32. The route's prefix length must be exactly 32 bits long, and any value is acceptable in the second byte and the fourth byte.

Evaluation of an Address Mask Match Type with Longest-Match Lookup

This example illustrates how a longest-match lookup evaluates a route filter that contains two `address-mask` match types. Consider the route filter configured in the routing policy term `term_3` below:

```
[edit]
policy-options {
    policy-statement from_customer_c {
        term term_3 {
            from {
                route-filter 10.0.1.0/24 address-mask 255.0.255.0;
                route-filter 10.0.2.0/24 address-mask 255.240.255.0;
            }
            then {
                ...
            }
        }
    }
}
```

Suppose that the incoming IPv4 route source address 10.1.1.0/24 is tested against the route filter configured in the policy term `term_3`:

1. The longest-match lookup tree for routing policy term `term_3` contains two match prefixes: one prefix for `10.0.1.0/24 address-mask 255.0.255.0` and one prefix for `10.0.2.0/24 address-mask 255.240.255.0`. When searching the tree for the longest-prefix match for a candidate, the longest-match lookup considers the number of contiguous high-order bits in the configured *netmask-value* instead of the length of the configured *destination-prefix*:

   - For the first route filter match criteria, the longest-match lookup entry is 10.0.0.0/8 because the netmask value contains 8 contiguous high-order bits.
   - For second route filter match criteria, the longest-match lookup entry is 10.0.0.0/12 because the netmask value contains 12 contiguous high-order bits.

   For the candidate route address 10.1.1.0/24, the longest-match lookup returns the tree entry 10.0.0.0/12, which is corresponds to the route filter match criteria `10.0.2.0/24 address-mask 255.240.255.0`.

2. Now that the longest-match prefix in `term_3` has been identified for the candidate route address, the candidate route address is evaluated against the route filter match criteria `10.0.2.0/24 address-mask 255.240.255.0`:

   a. To test the incoming IPv4 route address 10.1.1.0/24, the netmask value 255.240.255.0 is applied to 10.1.1.0/24. The result is 10.0.1.0.

   b. To test the configured destination prefix address 10.0.2.0/24, the netmask value 255.240.255.0 is applied to 10.0.2.0/24. The result is 10.0.2.0.

   c. Because the results are different, the route filter match fails. No actions, whether specified with the match criteria or with the `then` statement, are taken. The incoming IPv4 route address is not evaluated against any other match criteria.

RELATED DOCUMENTATION

Walkup for Route Filters Overview | 330

Example: Configuring Policy Chains and Route Filters | 258

Example: Configuring a Route Filter Policy to Specify Priority for Prefixes Learned Through OSPF | 362

Example: Configuring the MED Using Route Filters | 368

Understanding Route Filter and Source Address Filter Lists for Use in Routing Policy Match Conditions

Existing route filters and source address filters are configured and processed inline within the term of the policy statement. When route policies are changed, the entire policy is purged and rebuilt during the configuration parsing operation. When this happens on routing policies that include hundreds or even thousands of route filters and source address filters, a significant amount of time is added to the rebuild of the policy.

In order to speed the parsing operation, the `route-filter-list` and `source-address-filter-list` statements are available as another means of configuring route filters and source address filters. These statements maintain all the capabilities of the `route-filter` and `source-address-filter` statements, including consideration of the prefix length and match type of the individual prefixes in the list.

Route filters and route filter lists are typically used to match an incoming route address to destination match prefixes of any type except for unicast source addresses.

Source address filters and source address filter lists are typically used to match an incoming route address to unicast source addresses in Multiprotocol BGP (MBGP) and Multicast Source Discovery Protocol (MSDP) environments.

Multiple route filter lists and source address filter lists can be used within the same policy statements. Route filter lists and source address filter lists can also be used in conjunction with route filters and source address filters.

RELATED DOCUMENTATION

route-filter-list

Understanding Route Filters for Use in Routing Policy Match Conditions | 303

Understanding Load Balancing Using Source or Destination IP Only

In deep packet inspection (DPI) networks with per-subscriber awareness or transparent caches, all of the PE routers in the service provider network should route all traffic to and from a particular subscriber through the specific content server that maintains subscriber state for that subscriber. To reach the same server consistently, the traffic must be hashed onto the same link towards that specific server for traffic in both directions.

In order to accomplish this consistency, certain MX Series routers can be configured to make load-balancing decisions based solely on the source IP address or the destination IP address of the traffic.

From a service provider perspective, using only the source IP for inbound traffic, and the destination IP for outbound traffic limits the criteria used in hashing, making it more likely that a particular link will be chosen to forward the traffic.

NOTE: This feature will only work on IP-based traffic. In the case of L3VPN traffic, only MPLS lookup will be performed on the PE routers when the default label assignment scheme is used. In order to use source-or-destination only load-balancing with L3VPN, you can either configure `vrf-table-label` or add a vt- interface in the routing instance.

RELATED DOCUMENTATION

Configuring Load Balancing Using Source or Destination IP Only | 328

vrf-table-label

interface

Configuring Load Balancing Using Source or Destination IP Only

In equal-cost multipath, (ECMP) per-subscriber aware environments such as content service providers who service residential customers, traffic in both directions within the service provider network should always pass through the content servers that maintain the subscriber state information for a given subscriber. This is accomplished by calculating the load balancing hash based solely on source address for traffic coming into the service provider network and calculating the load balancing hash based solely on the destination address for traffic leaving the service provider network.

Source and destination only load balancing is generally configured in an ECMP or aggregated ethernet (AE) environment on an service provider network. It is usually applied to all of the PE routers. It is only supported for IPv4 (`inet`) and IPv6 (`inet6`) traffic.

You do not need any special configuration in place before starting this configuration.

NOTE: This feature will only work on IP-based traffic. In the case of L3VPN traffic, only MPLS lookup will be performed on the PE routers when the default label assignment scheme is used. In order to use source-or-destination only load-balancing with L3VPN, you can either configure `vrf-table-label` or add a vt- interface in the routing instance.

To configure load balancing using source or destination IP only, you first configure system-wide forwarding options with a prefix-length to use when calculating the hash-key. Then, you configure a

policy action of either `load-balance source-ip-only` or `load-balance destination-ip-only` within a policy statement.

1. To configure system-wide prefix length for use with source and destination IP only load balancing, insert the `source-destination-only-load-balancing` configuration statement at the [`edit forwarding-options enhanced-hash-key`] hierarchy level and add a prefix length:

```
[edit forwarding-options enhanced-hash-key]
source-destination-only-load-balancing {
    family inet {
        prefix-length prefix-length;
    }
    family inet6 {
        prefix-length prefix-length;
    }
}
```

2. To configure routing policy to use load balancing based on source or destination IP only, insert either the `source-ip-only` or `destination-ip-only` as an action statement within a policy statement at the [`edit policy-options policy-statement policy-name`] hierarchy level:

```
[edit policy-options policy-statement policy-name]
term term-name {
    from {
        route-filter filter-spec'
    }
    then {
        load-balance (source-ip-only | destination-ip-only);
    }
}
```

NOTE: The `source-ip-only` and `destination-ip-only` configuration elements cannot be used together in the same term. This is because of the directional nature of the traffic that we are load balancing. To use the two elements in the same policy statement, you create two separate terms, each using a route filter specification that addresses the same traffic. Then use `source-ip-only` for the inbound traffic and `destination-ip-only` for the outbound traffic.

NOTE:

RELATED DOCUMENTATION

Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms

Understanding Load Balancing Using Source or Destination IP Only | 327

Configuring Stateful Load Balancing on Aggregated Ethernet Interfaces

Walkup for Route Filters Overview

Use the walkup feature if you have concerns about policy performance because of split route filters across multiple policy terms. The walkup feature enables the consolidation of route filters under one policy term.

By default, Junos evaluates multiple route filters in a policy statement term by first finding the longest match prefix and then evaluating the conditions attached to the route filter, such as prefix range. If the route filter condition is false (for example, the prefix is not in the specified range), then the whole term is false, even if there are potentially true shorter route filter prefixes. Due to this behavior, there can be performance issues if route filters are split into individual policy statement terms. The walkup feature changes the default route filter behavior.

Some automated policy tools — for example, those used for autonomous system border routers in the Border Gateway Protocol (BGP) — break up route filters into multiple terms because of the default route filter behavior. Route filters are also used in routing protocols other than BGP; the walkup feature is not limited to BGP route filters.

NOTE: Technically, BGP does not deal with routes in the same way as OSPF or IS-IS. BGP "routes" are more properly called network layer reachability information (NLRI) updates. However, the term "route" is used in most documentation and is used here.

Route filters consist of three major parts:

1. A prefix and prefix length (for example, `10.0.0.0/8`)
2. A match condition (for example, `exact`)

3. An action that is carried out if both previous parts — the prefix and match condition — both evaluate to true (for example, `accept`)

So the `10.0.0.0/8 exact accept` route filter succeeds if and only if the prefix considered is `10.0.0.0/8` exactly. This route filter rejects routes with all other longer prefixes, such as `10.0.0.0/10`, although there might be other route filter terms in the policy chain that accept the `10.0.0.0/10` route.

NOTE: Although the `10.0.0.0/8` route and variations are not specifically reserved for documentation, the private RFC 1918 `10.0.0.0/8` address space is used in this topic because of the flexibility and realistic scenarios that this address spaces provides.

Route filters can be combined in a single policy statement term. In that case, evaluation becomes more complex. Consider the following routing policy:

```
[edit policy-options]
policy-statement RouteFilter-A {
    term RouteFilter-1 {
        from {
            route-filter 10.0.0.0/16 prefix-length-range /22-/24;
            route-filter 10.0.0.0/8 orlonger;
        }
        then accept;
    }
    term default {
        then reject;
    }
}
```

Note that the `10.0.0.0/8 orlonger` filter includes the `10.0.0.0/16 prefix-length-range /22-/24` filter in its scope. That is, any `10.0.0.0` route with a prefix of 8 bits or longer could also be a route with a prefix in the range between 22 and 24 bits.

By default, evaluation of a policy statement term with multiple route filters is a two-step process:

1. The policy framework software performs a longest-match lookup on the list based on prefix and prefix-length values.

2. The software considers the route filter condition (`orlonger`, `exact`, and so on). The route either fulfills the route filter condition (success) or does not match the route filter condition (failure).

Based on the results of these two steps, the action determined by the match or failure is applied to the route. In `Route-Filter-A`, this means that any route that is “true” is accepted and any route that is “false” in the `RouteFilter-1` term is rejected. This route becomes a hidden (filtered) route.

For example, consider what happens when the route `10.0.0.0/18` is evaluated by the policy statement `RouteFilter-A`:

First, the `10.0.0.0/18` route is evaluated by the `RouteFilter-1` term. Because `10.0.0.0/16` is longer than `10.0.0.0/8`, the `10.0.0.0/18` route matches the longer and more specific route prefix. Next, the match fails because the `10.0.0.0/18` route does not match the `prefix-length-range /22-/24` condition. So the route match fails in the `RouteFilter-1` term, and the policy examines the next term, the default term. The `10.0.0.0/18` route is rejected by the default term.

As a result, the `10.0.0.0/18` route is hidden (filtered). (The `10.0.0.0/18` route can still be found with the **show route hidden** command.)

The issue is that the user might actually want the `10.0.0.0/18` route to be accepted, not rejected. Naturally, a route filter with a `10.0.0.0/18 exact` configuration could be added. But in a backbone routing table with 100,000 or more entries, it is not possible to configure a route filter tuned to every possible route or every possible new route added to the network.

The default workaround to achieve the proper behavior from the example routing policy is to configure a separate term for each route filter. This is frequently done, as follows:

```
[edit policy-options]
policy-statement RouteFilter-A {
    term RouteFilter-1 {
        from {
            route-filter 10.0.0.0/16 prefix-length-range /22-/24;
        }
        then accept;
    }
    term RouteFilter-2 {
        from {
            route-filter 10.0.0.0/8 orlonger;
        }
        then accept;
    }
    term default {
        then reject;
    }
}
```

Now the 10.0.0.0/18 route is accepted because, although it still fails the RouteFilter-1 match condition, it matches the new RouteFilter-2 term (10.0.0.0/8 is the longest match, and the orlonger condition is true). The problem with this approach is that the complete routing policy now takes more time to evaluate than when multiple route filters are grouped. This method also makes maintenance more complex.

The issues with the one-term-per-route-filters approach are solved with the walkup statement and feature. Walkup alters the default behavior of route filter evaluation globally or on a per-policy basis.

The walkup feature allows terms with multiple route filters to “walk-up” the evaluation process to include less-specific routes as well as the longest match. In other words, the walkup knob changes the default behavior from “if one fails, then the term fails” to if “one matches, then the term matches.”

Consider the application of the walkup feature to the example policy statement (you can also apply walk-up globally to all policies configured):

```
[edit policy-options]
policy-statement RouteFilter-A {
    defaults {
        route-filter walkup;
    }
    term RouteFilter-1 {
        from {
            route-filter 10.0.0.0/16 prefix-length-range /22-/24;
            route-filter 10.0.0.0/8 orlonger;
        }
        then accept;
    }
    term default {
        then reject;
    }
}
```

This is what happens when the route prefix 10.0.0.0/18 is evaluated by the policy statement RouteFilter-A:

The default behavior is altered by the walkup knob. As before, the 10.0.0.0/18 route matches the longer and more specific route prefix because 10.0.0.0/16 is longer than 10.0.0.0/8. As before, this match fails because the 10.0.0.0/18 route does not match the prefix-length-range /22-/24 condition. However, this time the process continues by a “walk up” and examines the less specific 10.0.0.0/8 route filter. The route condition of orlonger matches this filter and therefore the route is accepted by the RouteFilter-1 term.

This can be verified (for a BGP route) by the **show route protocol bgp 10.0.0.0/18** command. This time, the route is not hidden.

If you enable the walkup feature globally, you can override it locally on a per-policy basis with the `[edit policy-options policy-statements policy-statement-name defaults route-filter no-walkup]` statement.

RELATED DOCUMENTATION

- Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency | 346
- Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency | 354
- Configuring Walkup for Route Filters to Improve Operational Efficiency | 334
- Route Filter Match Conditions | 72
- BGP Configuration Overview
- Verify That a Particular BGP Route Is Received on Your Router
- Example: Configuring BGP Route Advertisement

Configuring Walkup for Route Filters to Improve Operational Efficiency

Use the walkup feature if you have concerns about policy performance because of split route filters across multiple policy terms. The walkup feature enables the consolidation of route filters under one policy term.

If policy statements have been split into multiple terms because of the default route filter behavior, the route filter walkup feature allows you to consolidate multiple route filters into one policy statement term. By default, Junos OS evaluates multiple route filters in a policy statement term by first finding the longest match prefix and then evaluating the conditions attached to the route filter, such as the prefix range. If the route filter condition is false (for example, the prefix is not in the specified range), then the whole term is false, even if there are potentially true shorter route filter prefixes. The walkup feature alters this default behavior, locally or globally.

The route filter walkup feature is used anywhere multiple route filters are used in a policy statement. The walkup option is supported in the main routing instance at the `[edit policy-options]` hierarchy level and in logical systems at the `[edit logical-systems policy-options]` hierarchy level.

Before you begin configuring route filter walkup, be sure you have:

- A properly configured routing policy or set of routing policies
- A need to consolidate multiple route filter terms into fewer routing policy terms

Route filter walkup can be configured in two different ways. You can configure the `walkup` option globally at the `[edit policy-options default route-filter]` hierarchy level or in logical systems at the `[edit logical-systems policy-options default route-filter]` hierarchy level. When you configure the `walkup` option globally,

you alter the policy route filter behavior in every policy statement. Instead of the default policy statement behavior (if the longest match route filter is false, then the term is false), the `walkup` option changes this behavior globally (to “walk up” from the longest match route filter to less specific, and if any is true, then the term is true).

If you configure the `walkup` option globally, you can still override it locally on a per-routing-policy basis. So if you have enabled `walkup` globally, you can override it in a routing policy by configuring the `no-walkup` option statement at the `[edit policy-options policy-statement default route-filter]` hierarchy level. The `no-walkup` option restores the default route filter behavior locally for this policy statement.

NOTE: At the `[edit policy-options default route-filter]` global level, the only option is the `walkup` statement because the default behavior globally is “no walkup.” However, for an individual policy statement at the `[edit policy-options policy-statement default route-filter]` hierarchy level, you can configure either the `walkup` or `no-walkup` option statement. In this way, at the local level, you can control whether the policy statement performs a walkup (with the `walkup` statement configured) or no walkup (with the `no-walkup` statement configured. This gives the user maximum control over the `walkup` option

You configure the walkup feature globally with:

user@host> set policy-options defaults route-filter walkup

Alternatively, configure the walkup feature globally in a logical system with:

user@host> set logical-systems *logical-system-name* policy-options defaults route-filter walkup

You configure the walkup or no-walkup feature locally in a policy statement with:

user@host> set policy-options policy-statement *policy-statement-name* defaults route-filter [no-walkup | walkup]

Alternatively, configure the walkup feature locally in a logical system with:

user@host> set logical-systems *logical-system-name* policy-options policy-statement policy-statement-name defaults route-filter [no-walkup | walkup]

Route filter walkup behavior can be complex when the statements are configured at the global and local level at the same time. Table 16 on page 336 shows the behavior of a policy statement with all six possible combinations of the walkup option when you configure the feature both globally and locally.

Table 16: Route Filter Walkup and Policy Statements

Case:	Global Configuration	Local Configuration	Result
1	(none)	(none)	The device does not perform a walkup for any policy (default operation).
2	(none)	`walkup`	The device performs a walkup for this policy.
3	(none)	`no-walkup`	The device does not perform a walkup for any policy (default operation).
4	`walkup`	(none)	The device performs a walkup for all policies.
5	`walkup`	`walkup`	The device performs a walkup for all policies.
6	`walkup`	`no-walkup`	The device does not perform a walkup for this policy only.

Each row forms a possible use case numbered 1 through 6. Each walkup case is configured as follows:

- Case #1: This is a trivial configuration for backward compatibility. No route filter walkup is enabled either globally or locally. The device behaves exactly as it did before the feature was introduced. No route filter walkup occurs in any policy.
- Case #2: Route filter walkup is not enabled globally, but is enabled locally for a specific policy named `RouteFilter-Case2`. Route filter walkup occurs in this policy.

  To configure the route filter walkup locally for a specific policy:

  1. Enable the walkup feature locally for this policy statement.

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case2 defaults route-filter walkup
```

2. Configure policy terms locally (walkup applies to all terms in this policy).

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case2 term ...
```

3. Apply the policy statement to a routing protocol.

- Case #3: Route filter `walkup` is not enabled globally, but `no-walkup` is enabled locally for a specific policy named `RouteFilter-Case3`. (This case is not particularly helpful, because no walkup takes place in all policies by default, but does make local behavior explicit, even if walkup is enabled globally in the future.)

  To configure the route filter no-walkup locally for a specific policy:

  1. Enable the `no-walkup` feature locally for this policy statement.

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case3 defaults route-filter no-walkup
```

  2. Configure policy terms locally (`no-walkup` applies to this policy).

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case3 term ...
```

  3. Apply the policy statement to a routing protocol.

- Case #4: Route filter `walkup` is enabled globally, but not enabled locally for a specific policy named `RouteFilter-Case4`. Because of the global configuration, route filter `walkup` occurs in this policy.

  To configure the route filter walkup globally for a device:

  1. Enable the walkup feature globally for this device.

```
[edit policy-options]
user@host# set defaults route-filter walkup
```

NOTE: Global `walkup`, in contrast to the `walkup` or `no-walkup` statements configured locally in a policy statement, is configured at the `[edit policy-options defaults]` or `[edit logical-systems` *`logical-system-name`* `policy-options defaults]` hierarchy level and applies to all policies.

2. Configure policy statement `RouteFilter-Case4` and terms locally (walkup applies to this policy).

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case4 term ...
```

3. Apply the policy statement to a routing protocol.

- Case #5: Route filter `walkup` is enabled globally, and enabled locally for a specific policy named `RouteFilter-Case5`. Although this configuration might appear redundant (walkup enabled globally as well as locally), this ensures that route filter walkup occurs in this policy even if route filter walkup is deleted at the global level.

  To configure the route filter walkup globally for a device and locally for a specific policy:

  1. Enable the `walkup` feature globally for this device.

```
[edit policy-options]
user@host# set defaults route-filter walkup
```

> **NOTE**: Global walkup is configured at the `[edit policy-options defaults]` or `[edit logical-systems` *`logical-system-name`* `policy-options defaults]` hierarchy level and applies to all policies.

  2. Configure policy statement `RouteFilter-Case5` and enable `walkup` locally (`walkup` applies to this policy).

```
[edit policy-options]
user@host# set policy-statement Route-Filter-Case5 defaults route-filter walkup
```

  3. Configure policy statement `RouteFilter-Case5` and terms locally (walkup applies to this policy).

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case5 term ...
```

  4. Apply the policy statement to a routing protocol.

- Case #6: Route filter `walkup` is enabled globally, but overridden locally with `no-walkup` for a specific policy named `RouteFilter-Case6`. Because of the local configuration, no route filter walkup occurs in this policy. This case is useful to make sure that a local policy still functions exactly as before global walkup was enabled.

To configure the route filter walkup globally for a device and the no-walkup feature locally for a specific policy:

1. Enable the walkup feature globally for this device.

```
[edit policy-options]
user@host# set defaults route-filter walkup
```

> **NOTE**: Global walkup is configured at the `[edit policy-options defaults]` or `[edit logical-systems logical-system-name policy-options defaults]` hierarchy level and applies to all policies.

2. Configure policy statement `RouteFilter-Case6` and disable walkup locally with the `no-walkup` statement (no walkup is performed in this policy).

```
[edit policy-options]
user@host# set policy-statement Route-Filter-Case6 defaults route-filter walkup
```

3. Configure policy statement `RouteFilter-Case6` and terms locally.

```
[edit policy-options]
user@host# set policy-statement RouteFilter-Case6 term ...
```

4. Apply the policy statement to a routing protocol.

> **NOTE**: Keep in mind that a policy statement does nothing until it is applied as an import or export policy for the routing protocol itself. For BGP, this can be done at the global, group or neighbor level.

RELATED DOCUMENTATION

- Walkup for Route Filters Overview | 330
- Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency | 346
- Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency | 354
- Route Filter Match Conditions | 72

Verify That a Particular BGP Route Is Received on Your Router

Example: Configuring BGP Route Advertisement

Example: Configuring Route Filter Lists

IN THIS SECTION

- Requirements | 340
- Overview | 340
- Configuration | 341
- Verification | 343

Junos OS has long supported route filters for use in policy statements. Whenever policies are changed, the route filters have to be processed inline with the policy. Policies that contain large numbers of route filters take time to load.

This example shows how to create a route filter list and use that list in a policy statement. Route filter lists reduce the amount of time needed to reload a given policy.

NOTE: There is no speed benefit to using route filter lists in place of individual route filter entries when there are only a few route filters to process. The speed benefit is seen mainly in environments where there are hundreds or thousands of route filters listed within the policies.

Requirements

- A router configured with a routing protocol such as BGP or OSPF that is actively exchanging route information with its peers.
- The router that is configured with route filter lists must be running Junos OS Release 15.2 or later.

Overview

The `route-filter-list` statement allows for the creation of a pre-defined list of route filters for use in routing policies. You configure the list at the `[edit policy-options]` hierarchy level. The configured route filter list is then referenced as a match condition in the `from` section of a policy statement at the `[edit policy-options policy-statement` *`policy-statement-name`* `term` *`term-name`* `from]` hierarchy level.

In this example, the router that you are configuring is receiving some routes from its BGP neighbor 192.0.2.1. This is shown in the output of the `show route receive-protocol bgp 192.0.2.1` operational command.

```
user@router> show route receive-protocol bgp 192.0.2.1
inet.0: 17 destinations, 18 routes (16 active, 0 holddown, 1 hidden)
  Prefix                                  Nexthop           MED     Lclpref         AS path
* 198.151.100.0/29          192.0.2.1                            103 I
* 198.151.100.8/29          192.0.2.1                            103 I
* 203.0.113.0/29            192.0.2.1                            103 I
* 203.0.113.8/29            192.0.2.1                            103 I
* 203.0.113.16/29           192.0.2.1                            103 I
```

Configuration

IN THIS SECTION

- CLI Quick Configuration | 341
- Procedure | 342

CLI Quick Configuration

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

```
set policy-options route-filter-list rf-list-1 203.0.113.0/29 exact
set policy-options route-filter-list rf-list-1 203.0.113.8/29 exact
set policy-options route-filter-list rf-list-1 203.0.113.16/29 orlonger accept
set policy-options policy-statement rf-test-policy term term2 from route-filter 198.51.100.0/29
upto 198.51.100.0/30
set policy-options policy-statement rf-test-policy term term2 from route-filter 198.51.100.8/29
upto 198.51.100.8/30 accept
set policy-options policy-statement rf-test-policy term term2 from route-filter-list rf-list-1
set policy-options policy-statement rf-test-policy then reject
set protocols bgp group test-group import rf-test-policy
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

The following step-by-step procedure will lead you through the steps needed to:

- Configure a route filter list named `rf-list-1` and populate the list for later use in a route policy.
- Configure a routing policy statement named `rf-test-policy` that uses route filters and the configured route filter list.
- Configure BGP to use `rf-test-policy` as an import filter.

1. Configure a route filter list named `rf-list-1` for later use in a route policy.

```
[edit policy-options]
user@router# set route-filter-list rf-list-1
```

2. Populate the list `rf-list-1`.

   Note that one of the statements in the list has an action configured. This action will be carried out immediately upon a match with a received destination prefix.

```
[edit policy-options]
user@router# set route-filter-list rf-list-1 203.0.113.0/29 exact
user@router# set  route-filter-list rf-list-1 203.0.113.8/29 exact
user@router# set route-filter-list rf-list-1 203.0.113.16/29 orlonger accept
```

3. Configure a routing policy statement named `rf-test-policy` that uses route filters and the configured route filter list.

   The overall action for this policy is `reject`. There are individual route filters and elements of the route filter list that have a configured action of `accept`. The actions configured in the individual route filter statements and elements of the route filter list are carried out immediately upon matching a received destination prefix.

```
[edit policy-options]
user@router# set policy-statement rf-test-policy term term2 from route-filter 198.51.100.0/29
```

```
upto 198.51.100.0/30
user@router# set policy-statement rf-test-policy term term2 from route-filter 198.51.100.8/29
upto 198.51.100.8/30 accept
user@router# set policy-statement rf-test-policy term term2 from route-filter-list rf-list-1
user@router# set policy-statement rf-test-policy then reject
```

4. Configure BGP to use the configured policy as an import filter to selectively allow some routes and reject other routes from being added to the routing table.

```
[edit protocols bgp group test-group]
user@router# set import rf-test-policy
```

Verification

IN THIS SECTION

- Verifying the Configured Route Filter List | 343
- Verifying the Configured Policy Statement | 344
- Verifying That the Policy Statement Is Applied as an Import Policy in the BGP Protocol | 344
- Verifying That the Route Filter List Is Operating as Expected | 345

Verifying the Configured Route Filter List

Purpose

To confirm that the route filter list is properly configured, issue the `show policy-options route-filter-list` *`route-filter-list-name`* command at the `[edit]` hierarchy level.

Action

```
[edit]
user@routershow policy-options route-filter-list rf-list-1
203.0.113.0/29 exact;
203.0.113.8/29 exact;
203.0.113.16/29 orlonger accept;
```

Meaning

The output shows that the stored configuration is correct.

Verifying the Configured Policy Statement

Purpose

To confirm that the policy statement is properly configured, issue the `show policy-options policy-statement` *`policy-statement-name`* command at the `[edit]` hierarchy level.

Action

```
[edit]
user@router# show policy-options policy-statement rf-test-policy
from {
    route-filter 198.51.100.0/29 upto 198.51.100.0/30;
    route-filter 198.51.100.8/29 upto 198.51.100.8/30 accept;
     route-filter-list rf-list-1;
}
then reject;
```

Meaning

The output confirms that the stored configuration is correct.

Verifying That the Policy Statement Is Applied as an Import Policy in the BGP Protocol

Purpose

To confirm that the configured policy statement is applied as an import policy in the BGP Protocol, issue the `show protocols bgp import` command at the `[edit]` hierarchy level.

Action

```
[edit]
user@router# show protocols bgp import
import rf-test-policy;
```

Meaning

The outptut confirms that the stored configuration is correct.

If you have not already done so, you can issue the `commit` command at the `[edit]` hierarchy level so that the configuration is made active.

Verifying That the Route Filter List Is Operating as Expected

Purpose

Now that the configuration has been verified and committed, confirm the operation of the route filter list by issuing the `show route receive-protocol bgp 192.0.2.1` operational command.

Action

If you compare this output with the output of the same command issued prior to configuring the route filter list and policy statement, you see that some routes are no longer installed in the routing table.

```
user@router> show route receive-protocol bgp 192.0.2.1
inet.0: 14 destinations, 15 routes (13 active, 0 holddown, 1 hidden)
  Prefix                                        Nexthop           MED     Lclpref    AS path
* 198.151.100.8/29          192.0.2.1                            103 I
* 203.0.113.16/29           192.0.2.1                            103 I
```

Meaning

The output shows that three of the five previously installed BGP routes have been rejected by the policy statement `rf-test-policy`. The only routes that remain from the previous list are the two that had `accept` actions listed as part of the filter definition. The other routes were rejected by the action of the `policy-statement`.

RELATED DOCUMENTATION

route-filter-list

Understanding Route Filter and Source Address Filter Lists for Use in Routing Policy Match Conditions | 327

Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency

IN THIS SECTION

- Requirements | 346
- Overview | 347
- Configuring Route Filter Walkup Globally | 348
- Verification | 352
- Troubleshooting | 353

Use the walkup feature if you have concerns about policy performance because of split route filters across multiple policy terms. The walkup feature enables the consolidation of route filters under one policy term.

This example shows how to configure the route filter walkup feature globally for policy statements with route filters. When configured at the global level, the route filter walkup option applies to all policy statements. This example changes the default behavior of policy terms with multiple route filters globally, so that any reversion to the default “no walkup” behavior must be established locally.

Requirements

This example uses the following hardware and software components:

- A Juniper Networks router
- A Junos operating system from 13.3 or above

Before you configure route filter walkup locally, be sure you have:

- A properly configured routing policy or set of routing policies
- A need to consolidate multiple route filter terms into fewer routing policy terms

Overview

IN THIS SECTION

- Topology | 347

Routing protocols exchange information with other routers running the same routing protocols. In many cases, route filters are used in routing policy statements to filter prefixes for import or export. In some cases, when route filters are split into many separate terms, performance is impacted. The route filter walkup feature allows consolidation of policy statement terms for operational efficiency.

This example uses BGP, but the same walkup feature applies to any routing protocol that supports route filtering of input or output.

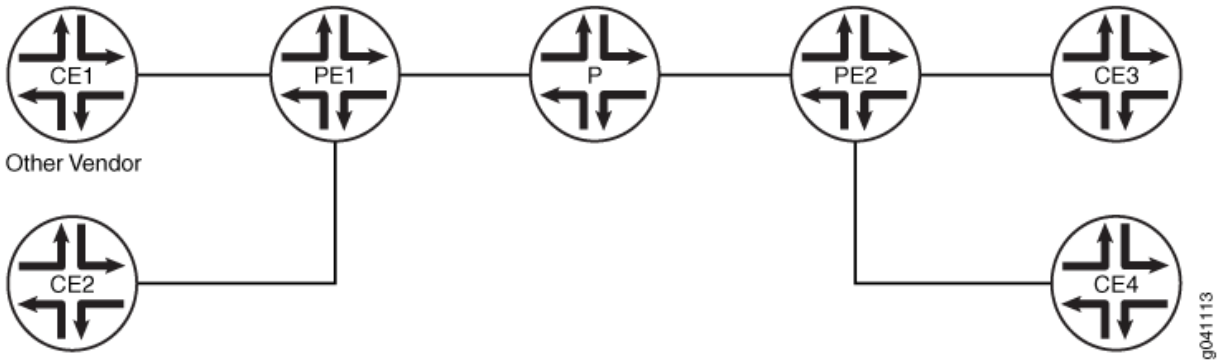
You can configure a Juniper Networks router to change the default operation of a term in a policy statement with route filters. By default, only a single longest match attempt is made for all route filters in a term. The walkup feature allows the router to “walk up” the route filters in a term from longest match to less specific in search of a true condition. This allows consolidation of multiple terms in a policy statement and corresponding operational efficiency.

This example changes the default behavior globally, for all policy statements. You can still configure `no-walkup` for an individual policy.

Topology

In the sample network in Figure 25 on page 348, the router CE1 is a router from another vendor. The rest are Juniper Networks routers. The walkup feature can be configured on any router in the figure, except for router CE1. The vendor of router CE1 might or not might support a similar feature.

Figure 25: Topology for the Global Walkup Example



In the example, the following addresses are used:

- 10.0.0.0/16
- 10.0.0.0/8

NOTE: Although the `10.0.0.0/8` address space is not specifically reserved for documentation, the private RFC 1918 `10.0.0.0/8` address space is used in this topic because of the flexibility and realistic scenarios that this address spaces provides.

Configuring Route Filter Walkup Globally

IN THIS SECTION

- CLI Quick Configuration | 348
- Procedure | 349
- Results | 350

CLI Quick Configuration

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

Device PE1

```
set policy-options defaults route-filter walkup
set policy-options policy-statement routeset1-import term prefixes1 from route-filter
10.0.0.0/16 prefix-length-range /22-/24
set policy-options policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/8
orlonger
set policy-options policy-statement routeset1-import term prefixes1 then accept
set policy-options policy-statement routeset1-import term reject-the-rest then reject
set policy-options policy-statement import-route-filter-a term import-routes from protocol bgp
set policy-options policy-statement import-route-filter-a term import-routes from policy
routeset1-import
set policy-options policy-statement import-route-filter-a term import-routes then next policy
set policy-options policy-statement import-route-filter-a term all-others then reject
set policy-options policy-statement route-filter-a-export term all then reject
set protocols bgp group routeset1 type external
set protocols bgp group routeset1 neighbor 10.0.10.13 import import-route-filter-a
set protocols bgp group routeset1 neighbor 10.0.10.13 family inet unicast
set protocols bgp group routeset1 neighbor 10.0.10.13 export route-filter-a-export
set protocols bgp group routeset1 neighbor 10.0.10.13 peer-as 64506
```

Procedure

Step-by-Step Procedure

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

To configure router PE1 to perform walkup globally and combine multiple route filters in one term:

1. Configure the walkup feature globally.

```
[edit policy-options defaults]
user@PE1# set route-filter walkup
```

2. Configure the policy statements for an import policy named `routeset1-import`.

```
[edit policy-options]
user@PE1# set policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/16
```

```
prefix-length-range /22-/24
user@PE1# set policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/8
orlonger
user@PE1# set policy-statement routeset1-import term prefixes1 then accept
user@PE1# set policy-statement routeset1-import term reject-the-rest then reject
```

3. Configure the policy options for the import and export policy statements.

```
[edit policy-options]
user@PE1# set policy-statement import-route-filter-a term import-routes from protocol bgp
user@PE1# set policy-statement import-route-filter-a term import-routes from policy routeset1-
import
user@PE1# set policy-statement import-route-filter-a term import-routes then next policy
user@PE1# set policy-statement route-filter-a-export term all-others then reject
```

4. Apply the import and export policies to a BGP neighbor.

```
[edit protocols bgp]
user@PE1# set group routeset1 type external
user@PE1# set group routeset1 neighbor 10.0.10.13 import import-route-filter-a
user@PE1# set group routeset1 neighbor 10.0.10.13 family inet unicast
user@PE1# set group routeset1 neighbor 10.0.10.13 export route-filter-a-export
user@PE1# set group routeset1 neighbor 10.0.10.13 peer-as 64506
```

Results

From configuration mode, confirm your configuration by entering the `show protocols` and `show policy-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options
    defaults {
        route-filter walkup;
    }

    policy-statement routeset1-import  {
    term prefixes1 {
        from {
            route-filter 10.0.0.0/16 prefix-length-range /22-/24;
```

```
            route-filter 10.0.0.0/8 orlonger;
        }
        then accept;
    }
    term reject-the-rest {
        then reject;
    }
}

policy-statement import-route-filter-a {
    term import-routes {
        from {
            protocol bgp;
            policy routeset1-import;
        }
        then next policy;
    }
    term all-others {
        then reject;
    }
}
policy-statement route-filter-a-export {
    term all {
        then reject;
    }
}
```

```
user@PE!# show protocols bgp
group routeset1 {
    type external;
    neighbor 10.0.10.13 {
        import import-route-filter-a;
        family inet {
            unicast;
        }
        export router-filter-a-export;
        peer-as 64506;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying Route Filter Operation | 352

Verifying Route Filter Operation

Purpose

Display expected information about the routes to confirm the route filters are working as expected.

Notice that the `10.0.0.0/8 orlonger` filter includes the `10.0.0.0/16 prefix-length-range /22-/24` filter in its scope. That is, any `10.0.0.0` route with a prefix of 8 bits or longer could also be a route with a prefix in the range between 22 and 24 bits. Without the walkup feature enabled, a route such as `10.0.0.0/16` would be rejected and become a hidden route. If the walkup feature is working as expected, then a route such as `10.0.0.0/16` would be accepted by the policy.

Action

From operational mode, enter the *show route protocol* `bgp 10.0.0.0/16` command. Make sure that `10.0.0.0/16` is not a hidden route.

```
user@PE1>show route protocol bgp 10.0.0.0/16
inet.0: 520762 destinations, 520764 routes (520760 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/16       *[BGP/170] 01:07:37, localpref 100
                        AS path: 64506, I, validation-state:  unverified
                      > to 10.0.100.13 via xe-0/2/0.0
```

As a further check, make sure that no routes that should be accepted are hidden routes. From operational mode, enter the *show route protocol* `bgp` *`ip-address-prefix`* `hidden` command to verify this.

Meaning

The presence of routes that are not the longest match in the configured policy route filter term shows that the walkup feature is functioning globally.

Troubleshooting

IN THIS SECTION

- Troubleshooting BGP | 353
- Troubleshooting Policy Statements | 353
- Troubleshooting Route Filters | 353

To troubleshoot route filter walkup globally:

Troubleshooting BGP

Problem

BGP is not functioning as expected.

Solution

See the BGP Configuration Overview topic, examples, and troubleshooting.

Troubleshooting Policy Statements

Problem

The policy statements are not functioning as expected.

Solution

See the Verify That a Particular BGP Route Is Received on Your Router and Example: Configuring BGP Route Advertisement topics, related examples, and troubleshooting.

Troubleshooting Route Filters

Problem

The route filters are not functioning as expected.

Solution

See the "Route Filter Match Conditions" on page 72 topic, examples, and troubleshooting.

RELATED DOCUMENTATION

- Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency | 354
- Walkup for Route Filters Overview | 330
- Configuring Walkup for Route Filters to Improve Operational Efficiency | 334
- Route Filter Match Conditions | 72
- BGP Configuration Overview
- Verify That a Particular BGP Route Is Received on Your Router
- Example: Configuring BGP Route Advertisement

Example: Configuring Walkup for Route Filters Locally to Improve Operational Efficiency

IN THIS SECTION

- Requirements | 355
- Overview | 355
- Configuring Route Filter Walkup Locally | 356
- Verification | 360
- Troubleshooting | 361

Use the walkup feature if you have concerns about policy performance because of split route filters across multiple policy terms. The walkup feature enables the consolidation of route filters under one policy term.

This example shows how to configure the route filter walkup feature locally for policy statements with route filters. When configured at the local level, the route filter walkup option applies only to the policy statement in which it is configured. This example does *not* change the default behavior of policy terms with route filters globally. This example establishes route filter walkup locally.

Requirements

This example uses the following hardware and software components:

- A Juniper Networks router
- A Junos operating system from 13.3 or above

Before you configure route filter walkup globally, be sure you have:

- A properly configured routing policy or set of routing policies
- A need to consolidate multiple route filter terms into fewer routing policy terms

Overview

IN THIS SECTION

- Topology | 355

Routing protocols exchange information with other routers running the same routing protocols. In many cases, route filters are used in routing policy statements to filter prefixes for import or export. In some cases, when route filters are split into many separate terms, performance is impacted. The route filter walkup feature allows consolidation of policy statement terms for operational efficiency.

This example uses BGP, but the same walkup feature applies to any routing protocol that supports route filtering of input or output.

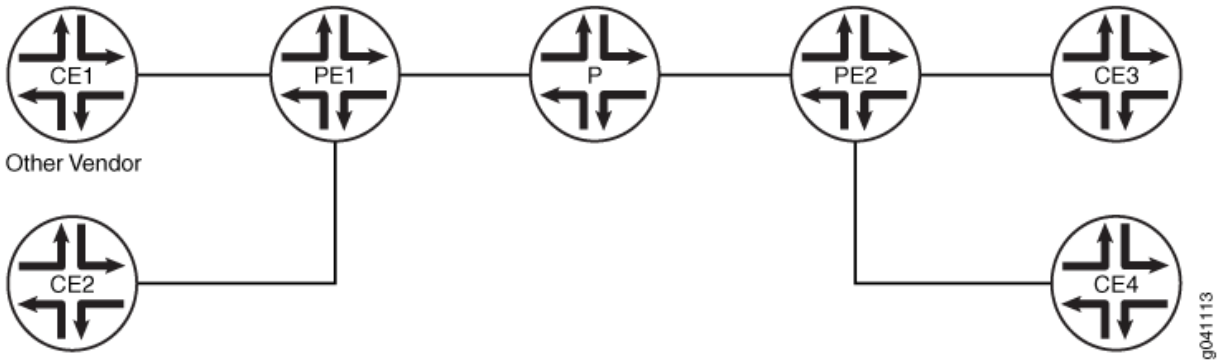
You can configure a Juniper Networks router to change the default operation of a term in a policy statement with route filters. By default, only a single longest match attempt is made for all route filters in a term. The walkup feature allows the router to “walk up” the route filters in a term from longest match to less specific in search of a true condition. This allows consolidation of multiple terms in a policy statement and corresponding operational efficiency.

This example changes the default behavior locally in a single policy statement. It does not affect the behavior of other policy statements.

Topology

In the sample network in Figure 26 on page 356, the router CE1 is a router from another vendor. The rest are Juniper Networks routers. The walkup feature can be configured on any router in the figure, except for router CE1. The vendor of router CE1 might or not might support a similar feature.

Figure 26: Topology for the Local Walkup Example



In the example, the following addresses are used:

- 10.0.0.0/16
- 10.0.0.0/8

NOTE: Although the `10.0.0.0/8` address space is not specifically reserved for documentation, the private RFC 1918 `10.0.0.0/8` address space is used in this topic because of the flexibility and realistic scenarios that this address spaces provides.

Configuring Route Filter Walkup Locally

IN THIS SECTION

- CLI Quick Configuration | **356**
- Procedure | **357**
- Results | **358**

CLI Quick Configuration

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

Device PE1

```
set policy-options policy-statement routeset1-import defaults route-filter walkup
set policy-options policy-statement routeset1-import term prefixes1 from route-filter
10.0.0.0/16 prefix-length-range /22-/24
set policy-options policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/8
orlonger
set policy-options policy-statement routeset1-import term prefixes1 then accept
set policy-options policy-statement routeset1-import term reject-the-rest then reject
set policy-options policy-statement import-route-filter-a term import-routes from protocol bgp
set policy-options policy-statement import-route-filter-a term import-routes from policy
routeset1-import
set policy-options policy-statement import-route-filter-a term import-routes then next policy
set policy-options policy-statement import-route-filter-a term all-others then reject
set policy-options policy-statement route-filter-a-export term all then reject
set protocols bgp group routeset1 type external
set protocols bgp group routeset1 neighbor 10.0.10.13 import import-route-filter-a
set protocols bgp group routeset1 neighbor 10.0.10.13 family inet unicast
set protocols bgp group routeset1 neighbor 10.0.10.13 export route-filter-a-export
set protocols bgp group routeset1 neighbor 10.0.10.13 peer-as 64506
```

Procedure

Step-by-Step Procedure

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

To configure router PE1 to perform walkup locally for multiple route filters in one term:

1. Configure the walkup feature locally in a policy named `routeset1-import`.

```
[edit policy-options policy-statement routeset1-import defaults]
user@PE1# set route-filter walkup
```

2. Configure the policy statements for an import policy named `routeset1-import`.

```
[edit policy-options ]
user@PE1# set policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/16
```

```
prefix-length-range /22-/24
user@PE1# set policy-statement routeset1-import term prefixes1 from route-filter 10.0.0.0/8
orlonger
user@PE1# set policy-statement routeset1-import term prefixes1 then accept
user@PE1# set policy-statement routeset1-import term reject-the-rest then reject
```

3. Configure the policy options for the import and export policy statements.

```
[edit policy-options]
user@PE1# set policy-statement import-route-filter-a term import-routes from protocol bgp
user@PE1# set policy-statement import-route-filter-a term import-routes from policy routeset1-
import
user@PE1# set policy-statement import-route-filter-a term import-routes then next policy
user@PE1# set policy-statement route-filter-a-export term all-others then reject
```

4. Apply the import and export policies to a BGP neighbor.

```
[edit protocols bgp]
user@PE1# set group routeset1 type external
user@PE1# set group routeset1 neighbor 10.0.10.13 import import-route-filter-a
user@PE1# set group routeset1 neighbor 10.0.10.13 family inet unicast
user@PE1# set group routeset1 neighbor 10.0.10.13 export route-filter-a-export
user@PE1# set group routeset1 neighbor 10.0.10.13 peer-as 64506
```

Results

From configuration mode, confirm your configuration by entering the `show protocols` and `show policy-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show policy-options
policy-statement routeset1-import  {
    defaults {
        route-filter walkup;
    }
    term prefixes1 {
        from {
            route-filter 10.0.0.0/16 prefix-length-range /22-/24;
            route-filter 10.0.0.0/8 orlonger;
```

```
        }
        then accept;
    }
    term reject-the-rest {
        then reject;
    }
}

policy-statement import-route-filter-a {
    term import-routes {
        from {
            protocol bgp;
            policy routeset1-import;
        }
        then next policy;
    }
    term all-others {
        then reject;
    }
}
policy-statement route-filter-a-export {
    term all {
        then reject;
    }
}
```

```
user@PE!# show protocols bgp
group routeset1 {
    type external;
    neighbor 10.0.10.13 {
        import import-route-filter-a;
        family inet {
            unicast;
        }
        export router-filter-a-export;
        peer-as 64506;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying Route Filter Operation | 360

Verifying Route Filter Operation

Purpose

Display expected information about the routes to confirm the route filters are working as expected.

Notice that the `10.0.0.0/8 orlonger` filter includes the `10.0.0.0/16 prefix-length-range /22-/24` filter in its scope. That is, any `10.0.0.0` route with a prefix of 8 bits or longer could also be a route with a prefix in the range between 22 and 24 bits. Without the walkup feature enabled in the policy example given, a route such as `10.0.0.0/16` would be rejected and become a hidden route. If the walkup feature is working as expected, then a route such as `10.0.0.0/16` would be accepted by the policy.

Action

From operational mode, enter the *show route protocol* `bgp 10.0.0.0/16` command. Make sure that `10.0.0.0/16` is not a hidden route.

```
user@PE1>show route protocol bgp 10.0.0.0/16
inet.0: 520762 destinations, 520764 routes (520760 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/16       *[BGP/170] 01:07:37, localpref 100
                        AS path: 64506, I, validation-state:  unverified
                      > to 10.0.100.13 via xe-0/2/0.0
```

As a further check, make sure that no routes that should be accepted are hidden routes. From operational mode, enter the *show route protocol* `bgp` *`ip-address-prefix`* `hidden` command to verify this.

Meaning

The presence of routes that are not the longest match in the configured policy route filter term shows that the walkup feature is functioning locally.

Troubleshooting

IN THIS SECTION

- Troubleshooting BGP | 361
- Troubleshooting Policy Statements | 361
- Troubleshooting Route Filters | 361

To troubleshoot route filter walkup locally:

Troubleshooting BGP

Problem

BGP is not functioning as expected.

Solution

See the BGP Configuration Overview topic, examples, and troubleshooting.

Troubleshooting Policy Statements

Problem

The policy statements are not functioning as expected.

Solution

See the Verify That a Particular BGP Route Is Received on Your Router and Example: Configuring BGP Route Advertisement topics, related examples, and troubleshooting.

Troubleshooting Route Filters

Problem

The route filters are not functioning as expected.

Solution

See the "Route Filter Match Conditions" on page 72 topic, examples, and troubleshooting.

RELATED DOCUMENTATION

- Example: Configuring Walkup for Route Filters Globally to Improve Operational Efficiency | 346
- Walkup for Route Filters Overview | 330
- Configuring Walkup for Route Filters to Improve Operational Efficiency | 334
- Route Filter Match Conditions | 72
- BGP Configuration Overview
- Verify That a Particular BGP Route Is Received on Your Router
- Example: Configuring BGP Route Advertisement

Example: Configuring a Route Filter Policy to Specify Priority for Prefixes Learned Through OSPF

IN THIS SECTION

- Requirements | 362
- Overview | 363
- Configuration | 364
- Verification | 368

This example shows how to create an OSPF import policy that prioritizes specific prefixes learned through OSPF.

Requirements

Before you begin:

- Configure the device interfaces. See the Interfaces User Guide for Security Devices.
- Configure the router identifiers for the devices in your OSPF network. See *Example: Configuring an OSPF Router Identifier*.

- Control OSPF designated router election See *Example: Controlling OSPF Designated Router Election*
- Configure a single-area OSPF network. See *Example: Configuring a Single-Area OSPF Network* .
- Configure a multiarea OSPF network. See *Example: Configuring a Multiarea OSPF Network*.

Overview

IN THIS SECTION

- Topology | **364**

In a network with a large number of OSPF routes, it can be useful to control the order in which routes are updated in response to a network topology change. In Junos OS Release 9.3 and later, you can specify a priority of high, medium, or low for prefixes included in an OSPF import policy. In the event of an OSPF topology change, high priority prefixes are updated in the routing table first, followed by medium and then low priority prefixes.

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. By default, such a route is now installed in the routing table with a priority of low. This behavior prevents traffic black holes, that is, silently discarded traffic, by ensuring consistent routing within the OSPF domain.

In general, OSPF routes that are not explicitly assigned a priority are treated as priority medium, except for the following:

- Summary discard routes have a default priority of low.
- Local routes that are not added to the routing table are assigned a priority of low.
- External routes that are rejected by import policy and thus not added to the routing table are assigned a priority of low.

Any available match criteria applicable to OSPF routes can be used to determine the priority. Two of the most commonly used match criteria for OSPF are the `route-filter` and `tag` statements.

In this example, the routing device is in area 0.0.0.0, with interfaces `fe-0/1/0` and `fe-1/1/0` connecting to neighboring devices. You configure an import routing policy named ospf-import to specify a priority for prefixes learned through OSPF. Routes associated with these prefixes are installed in the routing table in the order of the prefixes' specified priority. Routes matching `192.0.2.0/24 orlonger` are installed first because they have a priority of `high`. Routes matching `198.51.100.0/24 orlonger` are installed next because

they have a priority of `medium`. Routes matching `203.0.113.0/24 orlonger` are installed last because they have a priority of `low`. You then apply the import policy to OSPF.

NOTE: The priority value takes effect when a new route is installed, or when there is a change to an existing route.

Topology

Configuration

IN THIS SECTION

- CLI Quick Configuration | 364
- Procedure | 365

CLI Quick Configuration

To quickly configure an OSPF import policy that prioritizes specific prefixes learned through OSPF, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```
[edit]
set interfaces fe-0/1/0 unit 0 family inet address 192.168.8.4/30
set interfaces fe-0/1/0 unit 0 family inet address 192.168.8.5/30
set policy-options policy-statement ospf-import term t1 from route-filter 203.0.113.0/24 orlonger
set policy-options policy-statement ospf-import term t1 then priority low
set policy-options policy-statement ospf-import term t1 then accept
set policy-options policy-statement ospf-import term t2 from route-filter 198.51.100.0/24
orlonger
set policy-options policy-statement ospf-import term t2 then priority medium
set policy-options policy-statement ospf-import term t2 then accept
set policy-options policy-statement ospf-import term t3 from route-filter 192.0.2.0/24 orlonger
set policy-options policy-statement ospf-import term t3 then priority high
set policy-options policy-statement ospf-import term t3 then accept
set protocols ospf import ospf-import
```

```
set protocols ospf area 0.0.0.0 interface fe-0/1/0
set protocols ospf area 0.0.0.0 interface fe-1/1/0
```

Procedure

Step-by-Step Procedure

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

To configure an OSPF import policy that prioritizes specific prefixes:

1. Configure the interfaces.

```
[edit]
user@host# set interfaces fe-0/1/0 unit 0 family inet address 192.168.8.4/30
user@host# set interfaces fe-0/2/0 unit 0 family inet address 192.168.8.5/30
```

2. Enable OSPF on the interfaces.

> **NOTE**: For OSPFv3, include the `ospf3` statement at the `[edit protocols]` hierarchy level.

```
[edit]
user@host# set protocols ospf area 0.0.0.0 interface fe-0/1/0
user@host# set protocols ospf area 0.0.0.0 interface fe-0/2/0
```

3. Configure the policy to specify the priority for prefixes learned through OSPF.

```
[edit ]
user@host# set policy-options policy-statement ospf-import term t1 from route-filter
203.0.113.0/24 orlonger
user@host# set policy-options policy-statement ospf-import term t1 then priority low
user@host# set policy-options policy-statement ospf-import term t1 then accept
user@host# set policy-options policy-statement ospf-import term t2 from route-filter
198.51.100.0/24 orlonger
user@host# set policy-options policy-statement ospf-import term t2 then priority medium
user@host# set policy-options policy-statement ospf-import term t2 then accept
user@host# set policy-options policy-statement ospf-import term t3 from route-filter
```

```
192.0.2.0/24 orlonger
user@host# set policy-options policy-statement ospf-import term t3 then priority high
user@host# set policy-options policy-statement ospf-import term t3 then accept
```

4. Apply the policy to OSPF.

```
[edit]
user@host# set protocols ospf import ospf-import
```

5. If you are done configuring the device, commit the configuration.

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

Results

Confirm your configuration by entering the `show interfaces`, `show policy-options`, and the `show protocols ospf` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show interfaces
fe-0/1/0 {
    unit 0 {
        family inet {
            address 192.168.8.4/30;
        }
    }
}
    fe-0/2/0 {
        unit 0 {
            family inet {
                address 192.168.8.5/30;
            }
        }
    }
```

```
user@host# show protocols ospf
import ospf-import;
```

```
area 0.0.0.0 {
    interface fe-0/1/0.0;
    interface fe-0/2/0.0;
}
```

```
user@host# show policy-options
policy-statement ospf-import {
    term t1 {
        from {
            route-filter 203.0.113.0/24 orlonger;
        }
        then {
            priority low;
            accept;
        }
    }
    term t2 {
        from {
            route-filter 198.51.100.0/24 orlonger;
        }
        then {
            priority medium;
            accept;
        }
    }
    term t3 {
        from {
            route-filter 192.0.2.0/24 orlonger;
        }
        then {
            priority high;
            accept;
        }
    }
}
```

```
user@host# show protocols ospf
import ospf-import;
area 0.0.0.0 {
    interface fe-0/1/0.0;
```

```
    interface fe-0/2/0.0;
}
```

To confirm your OSPFv3 configuration, enter the `show interfaces`, `show policy-options`, and `show protocols ospf3` commands.

Verification

IN THIS SECTION

- Verifying the Prefix Priority in the OSPF Routing Table | **368**

Confirm that the configuration is working properly.

Verifying the Prefix Priority in the OSPF Routing Table

Purpose

Verify the priority assigned to the prefix in the OSPF routing table.

Action

From operational mode, enter the `show ospf route detail` for OSPFv2, and enter the `show ospf3 route detail` command for OSPFv3.

Example: Configuring the MED Using Route Filters

IN THIS SECTION

- Requirements | **369**
- Overview | **369**
- Configuration | **370**

Verification | 385

This example shows how to configure a policy that uses route filters to modify the multiple exit discriminator (MED) metric to advertise in BGP update messages.

Requirements

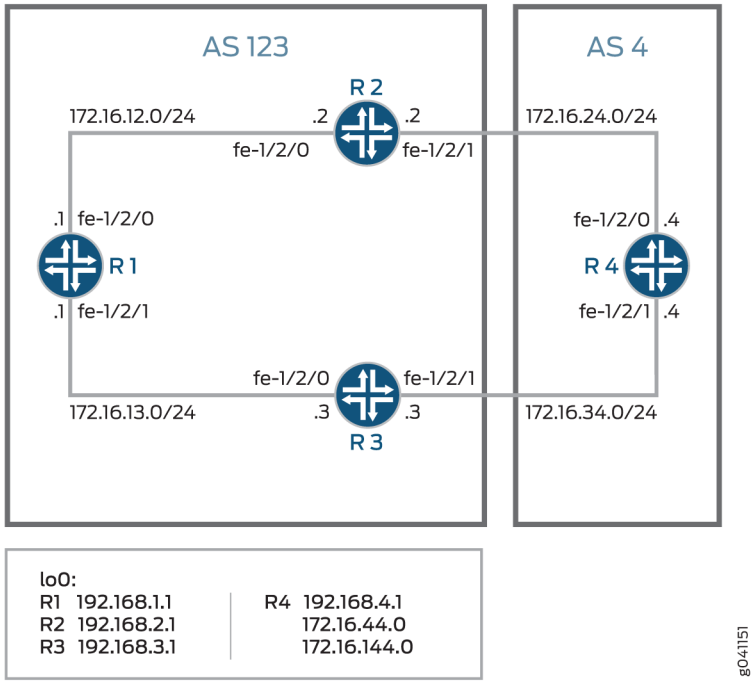
No special configuration beyond device initialization is required before you configure this example.

Overview

To configure a route-filter policy that modifies the advertised MED metric in BGP update messages, include the `metric` statement in the policy action.

Figure 27 on page 369 shows a typical network with internal peer sessions and multiple exit points to a neighboring autonomous system (AS).

Figure 27: Typical Network with IBGP Sessions and Multiple Exit Points



Device R4 has multiple loopback interfaces configured to simulate advertised prefixes. The extra loopback interface addresses are 172.16.44.0/32 and 172.16.144.0/32. This example shows how to configure Device R4 to advertise a MED value of 30 to Device R3 for all routes except 172.16.144.0.

For 172.16.144.0, a MED value of 10 is advertised to Device 3. A MED value of 20 is advertised to Device R2, regardless of the route prefix.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 370
- Configuring Device R1 | 372
- Configuring Device R2 | 375
- Configuring Device R3 | 378
- Configuring Device R4 | 382

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 1 family inet address 172.16.12.1/24
set interfaces fe-1/2/1 unit 2 family inet address 172.16.13.1/24
set interfaces lo0 unit 1 family inet address 192.168.1.1/32
set protocols bgp group internal type internal
set protocols bgp group internal local-address 192.168.1.1
set protocols bgp group internal export send-direct
set protocols bgp group internal neighbor 192.168.2.1
set protocols bgp group internal neighbor 192.168.3.1
set protocols ospf area 0.0.0.0 interface lo0.1 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.1
set protocols ospf area 0.0.0.0 interface fe-1/2/1.2
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 123
set routing-options router-id 192.168.1.1
```

Device R2

```
set interfaces fe-1/2/0 unit 3 family inet address 172.16.12.2/24
set interfaces fe-1/2/1 unit 4 family inet address 172.16.24.2/24
set interfaces lo0 unit 2 family inet address 192.168.2.1/32
set protocols bgp group internal type internal
set protocols bgp group internal local-address 192.168.2.1
set protocols bgp group internal export send-direct
set protocols bgp group internal neighbor 192.168.1.1
set protocols bgp group internal neighbor 192.168.3.1
set protocols bgp group external type external
set protocols bgp group external export send-direct
set protocols bgp group external peer-as 4
set protocols bgp group external neighbor 172.16.24.4
set protocols ospf area 0.0.0.0 interface lo0.2 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.3
set protocols ospf area 0.0.0.0 interface fe-1/2/1.4
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 123
set routing-options router-id 192.168.2.1
```

Device R3

```
set interfaces fe-1/2/0 unit 5 family inet address 172.16.13.3/24
set interfaces fe-1/2/1 unit 6 family inet address 172.16.34.3/24
set interfaces lo0 unit 3 family inet address 192.168.3.1/32
set protocols bgp group internal type internal
set protocols bgp group internal local-address 192.168.3.1
set protocols bgp group internal export send-direct
set protocols bgp group internal neighbor 192.168.1.1
set protocols bgp group internal neighbor 192.168.2.1
set protocols bgp group external type external
set protocols bgp group external export send-direct
set protocols bgp group external peer-as 4
set protocols bgp group external neighbor 172.16.34.4
set protocols ospf area 0.0.0.0 interface lo0.3 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.5
set protocols ospf area 0.0.0.0 interface fe-1/2/1.6
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
```

```
set routing-options autonomous-system 123
set routing-options router-id 192.168.3.1
```

Device R4

```
set interfaces fe-1/2/0 unit 7 family inet address 172.16.24.4/24
set interfaces fe-1/2/1 unit 8 family inet address 172.16.34.4/24
set interfaces lo0 unit 4 family inet address 192.168.4.1/32
set interfaces lo0 unit 4 family inet address 172.16.44.0/32
set interfaces lo0 unit 4 family inet address 172.16.144.0/32
set protocols bgp group external type external
set protocols bgp group external export send-direct
set protocols bgp group external peer-as 123
set protocols bgp group external neighbor 172.16.34.3 export med-10
set protocols bgp group external neighbor 172.16.34.3 export med-30
set protocols bgp group external neighbor 172.16.24.2 metric-out 20
set policy-options policy-statement med-10 from route-filter 172.16.144.0/32 exact
set policy-options policy-statement med-10 then metric 10
set policy-options policy-statement med-10 then accept
set policy-options policy-statement med-30 from route-filter 0.0.0.0/0 longer
set policy-options policy-statement med-30 then metric 30
set policy-options policy-statement med-30 then accept
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 4
set routing-options router-id 192.168.4.1
```

Configuring Device R1

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 1]
user@R1# set family inet address 172.16.12.1/24
```

```
[edit interfaces fe-1/2/1 unit 2]
user@R1# set family inet address 172.16.13.1/24
[edit interfaces lo0 unit 1]
user@R1# set family inet address 192.168.1.1/32
```

2. Configure BGP.

```
[edit protocols bgp group internal]
user@R1# set type internal
user@R1# set local-address 192.168.1.1
user@R1# set export send-direct
user@R1# set neighbor 192.168.2.1
user@R1# set neighbor 192.168.3.1
```

3. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R1# set interface lo0.1 passive
user@R1# set interface fe-1/2/0.1
user@R1# set interface fe-1/2/1.2
```

4. Configure a policy that accepts direct routes.

   Other useful options for this scenario might be to accept routes learned through OSPF or local routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R1# set from protocol direct
user@R1# set then accept
```

5. Configure the router ID and autonomous system (AS) number.

```
[edit routing-options]
user@R1# set autonomous-system 123
user@R1# set router-id 192.168.1.1
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
fe-1/2/0 {
    unit 1 {
        family inet {
            address 172.16.12.1/24;
        }
    }
}
fe-1/2/1 {
    unit 2 {
        family inet {
            address 172.16.13.1/24;
        }
    }
}
lo0 {
    unit 1 {
        family inet {
            address 192.168.1.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group internal {
        type internal;
        local-address 192.168.1.1;
        export send-direct;
        neighbor 192.168.2.1;
        neighbor 192.168.3.1;
    }
}
ospf {
    area 0.0.0.0 {
```

```
        interface lo0.1 {
            passive;
        }
        interface fe-1/2/0.1;
        interface fe-1/2/1.2;
    }
}
```

```
user@R1# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R1# show routing-options
autonomous-system 123;
router-id 192.168.1.1;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device R2

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 3]
user@R2# set family inet address 172.16.12.21/24
[edit interfaces fe-1/2/1 unit 4]
user@R2# set family inet address 172.16.24.2/24
```

```
[edit interfaces lo0 unit 2]
user@R2# set family inet address 192.168.2.1/32
```

2. Configure BGP.

```
[edit protocols bgp group internal]
user@R2# set type internal
user@R2# set local-address 192.168.2.1
user@R2# set export send-direct
user@R2# set neighbor 192.168.1.1
user@R2# set neighbor 192.168.3.1
[edit protocols bgp group external]
user@R2# set type external
user@R2# set export send-direct
user@R2# set peer-as 4
user@R2# set neighbor 172.16.24.4
```

3. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R2# set interface lo0.2 passive
user@R2# set interface fe-1/2/0.3
user@R2# set interface fe-1/2/1.4
```

4. Configure a policy that accepts direct routes.

   Other useful options for this scenario might be to accept routes learned through OSPF or local routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

5. Configure the router ID and autonomous system (AS) number.

```
[edit routing-options]
user@R2# set autonomous-system 123
user@R2# set router-id 192.168.2.1
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 3 {
        family inet {
            address 172.16.12.2/24;
        }
    }
}
fe-1/2/1 {
    unit 4 {
        family inet {
            address 172.16.24.2/24;
        }
    }
}
lo0 {
    unit 2 {
        family inet {
            address 192.168.2.1/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group internal {
        type internal;
        local-address 192.168.2.1;
        export send-direct;
        neighbor 192.168.1.1;
        neighbor 192.168.3.1;
    }
    group external {
        type external;
        export send-direct;
```

```
        peer-as 4;
        neighbor 172.16.24.4;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.2 {
            passive;
        }
        interface fe-1/2/0.3;
        interface fe-1/2/1.4;
    }
}
```

```
user@R2# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R2# show routing-options
autonomous-system 123;
router-id 192.168.2.1;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device R3

Step-by-Step Procedure

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

To configure Device R3:

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 5]
user@R3# set family inet address 172.16.13.3/24
[edit interfaces fe-1/2/1 unit 6]
user@R3# set family inet address 172.16.34.3/24
[edit interfaces lo0 unit 3]
user@R3# set family inet address 192.168.3.1/32
```

2. Configure BGP.

```
[edit protocols bgp group internal]
user@R3# set type internal
user@R3# set local-address 192.168.3.1
user@R3# set export send-direct
user@R3# set neighbor 192.168.1.1
user@R3# set neighbor 192.168.2.1
[edit protocols bgp group external]
user@R3# set type external
user@R3# set export send-direct
user@R3# set peer-as 4
user@R3# set neighbor 172.16.34.4
```

3. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R3# set interface lo0.3 passive
user@R3# set interface fe-1/2/0.5
user@R3# set interface fe-1/2/1.6
```

4. Configure a policy that accepts direct routes.

   Other useful options for this scenario might be to accept routes learned through OSPF or local routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R3# set from protocol direct
user@R3# set then accept
```

5. Configure the router ID and autonomous system (AS) number.

```
[edit routing-options]
user@R3# set autonomous-system 123
user@R3# set router-id 192.168.3.1
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
fe-1/2/0 {
    unit 5 {
        family inet {
            address 172.16.13.3/24;
        }
    }
}
fe-1/2/1 {
    unit 6 {
        family inet {
            address 172.16.34.3/24;
        }
    }
}
lo0 {
    unit 3 {
        family inet {
            address 192.168.3.1/32;
        }
    }
}
```

```
user@R3# show protocols
bgp {
    group internal {
        type internal;
```

```
        local-address 192.168.3.1;
        export send-direct;
        neighbor 192.168.1.1;
        neighbor 192.168.2.1;
    }
    group external {
        type external;
        export send-direct;
        peer-as 4;
        neighbor 172.16.34.4;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.3 {
            passive;
        }
        interface fe-1/2/0.5;
        interface fe-1/2/1.6;
    }
}
```

```
user@R3# show policy-options
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R3# show routing-options
autonomous-system 123;
router-id 192.168.3.1;
```

If you are done configuring the device, enter `commit` from configuration mode.

Configuring Device R4

Step-by-Step Procedure

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

To configure Device R4:

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 7]
user@R4# set family inet address 172.16.24.4/24
[edit interfaces fe-1/2/1 unit 8]
user@R4# set family inet address 172.16.34.4/24
[edit interfaces lo0 unit 4]
user@R4# set family inet address 192.168.4.1/32
user@R4# set family inet address 172.16.44.0/32
user@R4# set family inet address 172.16.144.0/32
```

   Device R4 has multiple loopback interface addresses to simulate advertised prefixes.

2. Configure a policy that accepts direct routes.

   Other useful options for this scenario might be to accept routes learned through OSPF or local routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R4# set from protocol direct
user@R4# set then accept
```

3. Configure BGP.

```
[edit protocols bgp group external]
user@R4# set type external
user@R4# set export send-direct
user@R4# set peer-as 123
```

4. Configure the two MED policies.

```
[edit policy-options]
set policy-statement med-10 from route-filter 172.16.144.0/32 exact
set policy-statement med-10 then metric 10
set policy-statement med-10 then accept
set policy-statement med-30 from route-filter 0.0.0.0/0 longer
set policy-statement med-30 then metric 30
set policy-statement med-30 then accept
```

5. Configure the two EBGP neighbors, applying the two MED policies to Device R3, and a MED value of 20 to Device R2.

```
[edit protocols bgp group external]
user@R4# set neighbor 172.16.34.3 export med-10
user@R4# set neighbor 172.16.34.3 export med-30
user@R4# set neighbor 172.16.24.2 metric-out 20
```

6. Configure the router ID and autonomous system (AS) number.

```
[edit routing-options]
user@R4# set autonomous-system 4
user@R4# set router-id 192.168.4.1
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R4# show interfaces
fe-1/2/0 {
    unit 7 {
        family inet {
            address 172.16.24.4/24;
        }
    }
}
```

```
fe-1/2/1 {
    unit 8 {
        family inet {
            address 172.16.34.4/24;
        }
    }
}
lo0 {
    unit 4 {
        family inet {
            address 192.168.4.1/32;
            address 172.16.44.0/32;
            address 172.16.144.0/32;
        }
    }
}
```

```
user@R4# show protocols
bgp {
    group external {
        type external;
        export send-direct;
        peer-as 123;
        neighbor 172.16.24.2 {
            metric-out 20;
        }
        neighbor 172.16.34.3 {
            export [ med-10 med-30 ];
        }
    }
}
```

```
user@R4# show policy-options
policy-statement med-10 {
    from {
        route-filter 172.16.144.0/32 exact;
    }
    then {
        metric 10;
        accept;
```

```
    }
}
policy-statement med-30 {
    from {
        route-filter 0.0.0.0/0 longer;
    }
    then {
        metric 30;
        accept;
    }
}
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R4# show routing-options
autonomous-system 4;
router-id 192.168.4.1;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Checking the Active Path from Device R1 to Device R4 | **385**
- Verifying That Device R4 Is Sending Its Routes Correctly | **387**

Confirm that the configuration is working properly.

Checking the Active Path from Device R1 to Device R4

Purpose

Verify that the active path goes through Device R2.

Action

From operational mode, enter the `show route protocol bgp` command.

```
user@R1> show route protocol bgp
inet.0: 13 destinations, 19 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.12.0/24       [BGP/170] 4d 01:13:32, localpref 100, from 192.168.2.1
                      AS path: I
                    > to 172.16.12.2 via fe-1/2/0.1
172.16.13.0/24       [BGP/170] 3d 05:36:10, localpref 100, from 192.168.3.1
                      AS path: I
                    > to 172.16.13.3 via fe-1/2/1.2
172.16.24.0/24       [BGP/170] 4d 01:13:32, localpref 100, from 192.168.2.1
                      AS path: I
                    > to 172.16.12.2 via fe-1/2/0.1
172.16.34.0/24       [BGP/170] 3d 05:36:10, localpref 100, from 192.168.3.1
                      AS path: I
                    > to 172.16.13.3 via fe-1/2/1.2
172.16.44.0/32     *[BGP/170] 00:06:03, MED 20, localpref 100, from 192.168.2.1
                      AS path: 4 I
                    > to 172.16.12.2 via fe-1/2/0.1
172.16.144.0/32 *[BGP/170] 00:06:03, MED 10, localpref 100, from 192.168.3.1
                      AS path: 4 I
                    > to 172.16.13.3 via fe-1/2/1.2
192.168.2.1/32      [BGP/170] 4d 01:13:32, localpref 100, from 192.168.2.1
                      AS path: I
                    > to 172.16.12.2 via fe-1/2/0.1
192.168.3.1/32      [BGP/170] 3d 05:36:10, localpref 100, from 192.168.3.1
                      AS path: I
                    > to 172.16.13.3 via fe-1/2/1.2
192.168.4.1/32     *[BGP/170] 00:06:03, MED 20, localpref 100, from 192.168.2.1
                      AS path: 4 I
                    > to 172.16.12.2 via fe-1/2/0.1
```

Meaning

The output shows that the preferred path to the routes advertised by Device R4 is through Device R2 for all routes except 172.16.144.0/32. For 172.16.144.0/32, the preferred path is through Device R3.

Verifying That Device R4 Is Sending Its Routes Correctly

Purpose

Make sure that Device R4 is sending update messages with a value of 20 to Device R2 and a value of 30 to Device R3.

Action

From operational mode, enter the `show route advertising-protocol bgp` command.

```
user@R4> show route advertising-protocol bgp 172.16.24.2
inet.0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.24.0/24           Self                 20                 I
* 172.16.34.0/24           Self                 20                 I
* 172.16.44.0/32           Self                 20                 I
* 172.16.144.0/32      Self                 20                 I
* 192.168.4.1/32           Self                 20                 I
```

```
user@R4> show route advertising-protocol bgp 172.16.34.3
inet.0: 11 destinations, 13 routes (11 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.24.0/24           Self                 30                 I
* 172.16.34.0/24           Self                 30                 I
* 172.16.44.0/32           Self                 30                 I
* 172.16.144.0/32      Self                 10                 I
* 192.168.4.1/32           Self                 30                 I
```

Meaning

The MED column shows that Device R4 is sending the correct MED values to its two EBGP neighbors.

RELATED DOCUMENTATION

- Example: Associating the MED Path Attribute with the IGP Metric and Delaying MED Updates
- Understanding Route Filters for Use in Routing Policy Match Conditions | **303**
- *Understanding BGP Path Selection*

Understanding External BGP Peering Sessions

Example: Configuring Layer 3 VPN Protocol Family Qualifiers for Route Filters

IN THIS SECTION

- Requirements | 388
- Overview | 388
- Configuration | 389
- Verification | 392

This example shows how to control the scope of BGP import policies by configuring a family qualifier for the BGP import policy. The family qualifier specifies routes of type `inet`, `inet6`, `inet-vpn`, or `inet6-vpn`.

Requirements

This example uses Junos OS Release 10.0 or later.

Before you begin:

- Configure the device interfaces.
- Configure an interior gateway protocol. See the Junos OS Routing Protocols Library.
- Configure a BGP session for multiple route types. For example, configure the session for both family `inet` routes and family `inet-vpn` routes. See *Configuring IBGP Sessions Between PE Routers in VPNs* and *Configuring Layer 3 VPNs to Carry IPv6 Traffic*.

Overview

Family qualifiers cause a route filter to match only one specific family. When you configure an IPv4 route filter without a family qualifier, as shown here, the route filter matches `inet` and `inet-vpn` routes.

```
route-filter ipv4-address/mask;
```

Likewise, when you configure an IPv6 route filter without a family qualifier, as shown here, the route filter matches `inet6` and `inet6-vpn` routes.

```
route-filter ipv6-address/mask;
```

Consider the case in which a BGP session has been configured for both family `inet` routes and family `inet-vpn` routes, and an import policy has been configured for this BGP session. This means that both family `inet` and family `inet-vpn` routes, when received, share the same import policy. The policy term might look as follows:

```
from {
    route-filter 0.0.0.0/0 exact;
}
then {
    next-hop self;
    accept;
}
```

This route-filter logic matches an `inet` route of 0.0.0.0 and an `inet-vpn` route whose IPv4 address portion is 0.0.0.0. The 8-byte route distinguisher portion of the `inet-vpn` route is not considered in the route-filter matching. This is a change in Junos OS behavior that was introduced in Junos OS Release 10.0.

If you do not want your policy to match both types of routes, add a family qualifier to your policy. To have the route-filter match only `inet` routes, add the family `inet` policy qualifier. To have the route-filter match only `inet-vpn` routes, add the family `inet-vpn` policy qualifier.

The family qualifier is evaluated before the route-filter is evaluated. Thus, the route-filter is not evaluated if the family match fails. The same logic applies to family `inet6` and family `inet6-vpn`. The route-filter used in the `inet6` example must use an IPv6 address. There is a potential efficiency gain in using a family qualifier because the family qualifier is tested before most other qualifiers, quickly eliminating routes from undesired families.

Configuration

IN THIS SECTION

- Procedure | 390
- Results | 391

Procedure

CLI Quick Configuration

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

inet Example

```
set policy-options policy-statement specific-family from family inet
set policy-options policy-statement specific-family from route-filter 0.0.0.0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Inet-vpn Example

```
set policy-options policy-statement specific-family from family inet-vpn
set policy-options policy-statement specific-family from route-filter 0.0.0.0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

inet6 Example

```
set policy-options policy-statement specific-family from family inet6
set policy-options policy-statement specific-family from route-filter 0::0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Inet6-vpn Example

```
set policy-options policy-statement specific-family from family inet6-vpn
set policy-options policy-statement specific-family from route-filter 0::0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Step-by-Step Procedure

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

To configure a flow map:

1. Configure the family qualifier.

```
[edit policy-options]
user@host# set policy-statement specific-family from family inet
```

2. Configure the route filter.

```
[edit policy-options]
user@host# set policy-statement specific-family from route-filter 0.0.0.0/0 exact
```

3. Configure the policy actions.

```
[edit policy-options]
user@host# set policy-statement specific-family then next-hop self
user@host# set policy-statement specific-family then accept
```

4. Apply the policy.

```
[edit protocols bgp]
user@host# set import specific-family
```

Results

From configuration mode, confirm your configuration by issuing the `show protocols` and `show policy-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show protocols
bgp {
    import specific-family;
```

```
}
user@host# show policy-options
policy-statement specific-family {
    from {
        family inet;
        route-filter 0.0.0.0/0 exact;
    }
    then {
        next-hop self;
        accept;
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Repeat the procedure for every protocol family for which you need a specific route-filter policy.

Verification

To verify the configuration, run the following commands:

- `show route advertising-protocol bgp neighbor detail`
- `show route instance instance-name detail`

Understanding Prefix Lists for Use in Routing Policy Match Conditions

IN THIS SECTION

- Configuring Prefix Lists | **394**
- How Prefix Lists Are Evaluated in Routing Policy Match Conditions | **395**
- Configuring Prefix List Filters | **395**

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

Suppose, for example, that you configure the following prefix list:

```
prefix-list bgp179 {
    apply-path "protocols bgp group <*> neighbor <*>";
}
```

This works well when all neighbors on the device are in the same address family.

When the neighbors are in different address families, for example when both IPv4 and IPv6 neighbors are configured, you can use a prefix list as follows:

```
prefix-list IPV4-BGP-NEIGHBORS {
    apply-path "protocols bgp group <*> neighbor <*.*.*.*>";
}
prefix-list IPV6-BGP-NEIGHBORS {
    apply-path "protocols bgp group <*> neighbor <*:*:*>";
}
```

One prefix list matches IPv4 addresses. The other matches IPv6 addresses. You can run the `show configuration policy-options prefix-list prefix-list name | display inheritance` command to verify 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 17 on page 393.

Table 17: Prefix List and Route List Differences

Feature	Prefix List	Route Lists
Action	Can specify action in a `then` statement only. If specified, the 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 "Understanding Route Filters for Use in Routing Policy Match Conditions" on page 303.

This section includes the following information:

Configuring Prefix Lists

You can create a named prefix list and include it in a routing policy with the `prefix-list` match condition (described in "Routing Policy Match Conditions" on page 58).

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

```
[edit policy-options]
    prefix-list prefix-list-name {
    apply-path path;
    ip-addresses;
}
```

You can use the `apply-path` statement to include all prefixes (and their associated network mask) 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 127 bytes 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*prefix-length*. 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. You cannot apply actions to individual prefixes in the list.

You can specify the same prefix list in the `from` statement of multiple routing policies or firewall filters. For information about firewall filters, see "Guidelines for Configuring Firewall Filters" on page 814 and "Guidelines for Applying Standard Firewall Filters" on page 821.

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.

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 Routing Policy Prefix Lists " on page 396.

How Prefix Lists Are Evaluated in Routing Policy Match Conditions

During prefix list evaluation, the policy framework software performs a *longest-match lookup*, which means that the software searches for the prefix in the list with the longest length. The order in which you specify the prefixes, from top to bottom, does not matter. The software then compares a route's source address to the longest prefix.

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

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

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 18 on page 396.

The *actions* option is the action to take if the prefix list matches. It can be one or more of the actions listed in "Configuring Flow Control Actions" on page 76 and "Configuring Actions That Manipulate Route Characteristics" on page 77.

Table 18: Route List Match Types for a Prefix List Filter

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

RELATED DOCUMENTATION

Example: Configuring Routing Policy Prefix Lists | 396

Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List | 1059

Example: Configuring Routing Policy Prefix Lists

IN THIS SECTION

- Requirements | 397
- Overview | 397
- Configuration | 400
- Verification | 408

In Junos OS, prefix lists provide one method of defining a set of routes. Junos OS provides other methods of accomplishing the same task, such as route filters. A prefix list is a listing of IP prefixes that represent a set of routes that are used as match criteria in an applied policy. Such a list might be useful for representing a list of customer routes in your autonomous system (AS). A prefix list is given a name and is configured within the `[edit policy-options]` configuration hierarchy.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Updated and re-validated using Junos OS Release 22.1R1.

Overview

IN THIS SECTION

- Topology | **399**

Prefix lists are similar to a list of route filters. The functional difference between route filters and prefix lists is that you cannot specify a range using a prefix list. You can simulate a range using a prefix list by including additional prefixes in the list, or by using two prefix lists, one shorter and one longer, setting one to accept and the other to reject. You can also filter a prefix list using the `prefix-list-filter` match condition. Your choices are `exact`, `longer`, and `orlonger`.

The benefit of a prefix list over a list of route filters is seen when the prefixes are referenced in several different locations. For instance, a prefix list can be referenced in a BGP import policy, an export policy, an RPF policy, in firewall filters, in loopback filters, in setting a multicast scope, and so on.

When your list of prefixes changes, rather than trying to remember the many different locations prefixes are configured, you can instead update the prefix list, changing the prefix one time instead of multiple times. This helps to reduce the likelihood of configuration errors, such as mistyping the address in a location or forgetting to update one or more locations.

Prefix lists also help when managing a large number of devices. You can write the various filters and policies as generically as possible, referencing prefix lists instead of specific IP addresses. The more complex logic in the filters and policies has to be written only one time, with minimal per-device and per-site customizations.

As shown in Figure 28 on page 400, each router in AS 65000 has customer routes. Device R1 assigns customer routes within the 172.16.1.0/24 subnet. Device R2 and Device R3 assign customer routes within the 172.16.2.0/24 and 172.16.3.0/24 subnets, respectively. Device R1 has been designated the

central point in AS 65000 to maintain a complete list of customer routes. Device R1 has a prefix list called `customers`, as follows:

```
user@R1# show policy-options
prefix-list customers {
    172.16.1.16/28;
    172.16.1.32/28;
    172.16.1.48/28;
    172.16.1.64/28;
    172.16.2.16/28;
    172.16.2.32/28;
    172.16.2.48/28;
    172.16.2.64/28;
    172.16.3.16/28;
    172.16.3.32/28;
    172.16.3.48/28;
    172.16.3.64/28;
}
```

As you can see, the prefix list does not contain a match type for each route (as you would see with a route filter). This is an important point when using a prefix list in a policy. Routes match only if they exactly match one of the prefixes in the list. In other words, each route in the list must appear in the routing table exactly as it is configured in the prefix list.

You reference the prefix list as a match criterion within a policy like this:

```
user@R1# show policy-options
policy-statement customer-routes {
    term get-routes {
        from {
            prefix-list customers;
        }
        then accept;
    }
    term others {
        then reject;
    }
}
```

In this example, all the routes in the `customers` prefix list appear in the routing table on Device R1. Device R2 and Device R3 export to Device R1 static routes to their customers.

As previously mentioned, you can use the `prefix-list-filter` match condition with the `exact`, `longer`, or `orlonger` match type. This provides a way to avoid the prefix list exact-match limitation of prefix lists. For example:

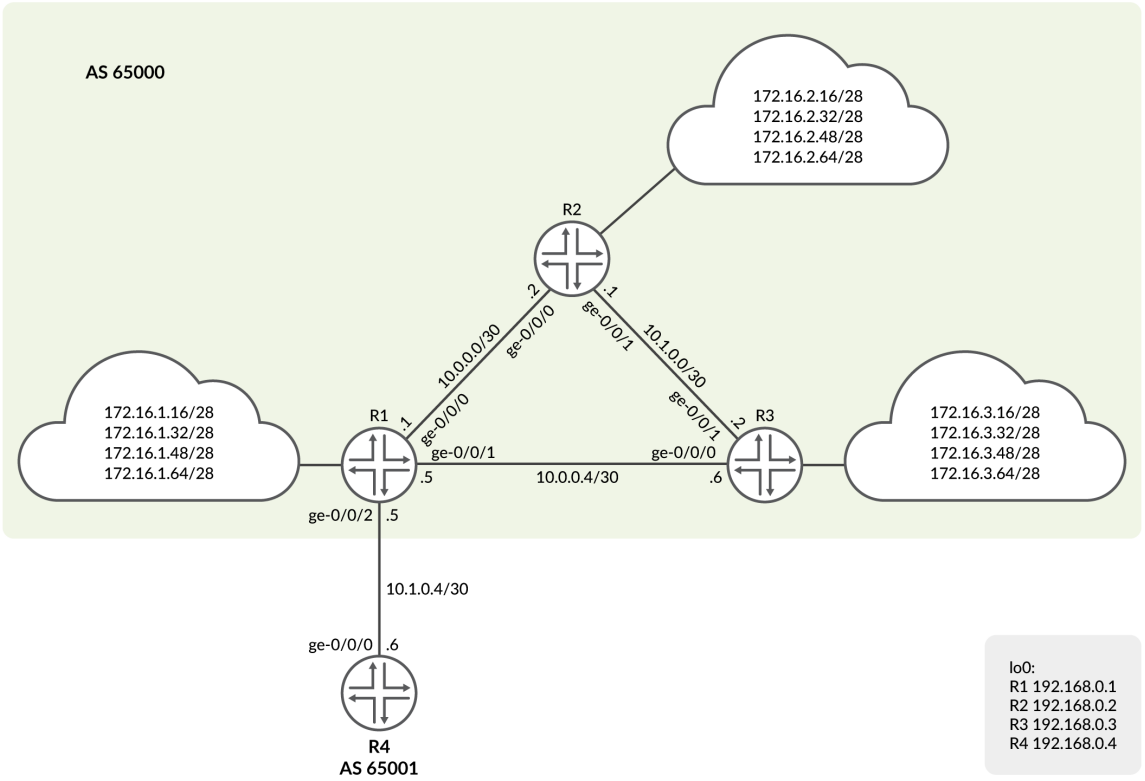
```
user@R1# show policy-options
policy-statement customer-routes {
    term get-routes {
        from {
            prefix-list-filter customers orlonger;
        }
        then accept;
    }
    term others {
        then reject;
    }
}
```

The example demonstrates the effects of both the `prefix-list` match condition and the `prefix-list-filter` match condition.

Topology

Figure 28 on page 400 shows the sample network.

Figure 28: BGP Topology for Policy Prefix Lists



"CLI Quick Configuration" on page 401 shows the configuration for all of the devices in Figure 28 on page 400.

The section "No Link Title" on page 403 describes the steps on Device R1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **401**
- Procedure | **403**

CLI Quick Configuration

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

Device R1

```
set interfaces ge-0/0/0 unit 0 description to_R2
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces ge-0/0/1 unit 0 description to_R3
set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.5/30
set interfaces ge-0/0/2 unit 0 description to_R4
set interfaces ge-0/0/2 unit 0 family inet address 10.1.0.5/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set policy-options prefix-list customers 172.16.1.16/28
set policy-options prefix-list customers 172.16.1.32/28
set policy-options prefix-list customers 172.16.1.48/28
set policy-options prefix-list customers 172.16.1.64/28
set policy-options prefix-list customers 172.16.2.16/28
set policy-options prefix-list customers 172.16.2.32/28
set policy-options prefix-list customers 172.16.2.48/28
set policy-options prefix-list customers 172.16.2.64/28
set policy-options prefix-list customers 172.16.3.16/28
set policy-options prefix-list customers 172.16.3.32/28
set policy-options prefix-list customers 172.16.3.48/28
set policy-options prefix-list customers 172.16.3.64/28
set policy-options policy-statement customer-routes term get-routes from prefix-list customers
set policy-options policy-statement customer-routes term get-routes then accept
set policy-options policy-statement customer-routes term others then reject
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 65000
set routing-options static route 172.16.1.16/28 discard
set routing-options static route 172.16.1.32/28 discard
set routing-options static route 172.16.1.48/28 discard
set routing-options static route 172.16.1.64/28 discard
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.1
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group to_65001 type external
set protocols bgp group to_65001 export customer-routes
```

```
set protocols bgp group to_65001 neighbor 10.1.0.6 peer-as 65001
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Device R2

```
set interfaces ge-0/0/0 unit 0 description to_R1
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.2/30
set interfaces ge-0/0/1 unit 0 description to_R3
set interfaces ge-0/0/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 65000
set routing-options static route 172.16.2.16/28 discard
set routing-options static route 172.16.2.32/28 discard
set routing-options static route 172.16.2.48/28 discard
set routing-options static route 172.16.2.64/28 discard
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int neighbor 192.168.0.1 export send-static
set protocols bgp group int neighbor 192.168.0.3
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Device R3

```
set interfaces ge-0/0/0 unit 0 description to_R1
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.6/30
set interfaces ge-0/0/1 unit 0 description to_R2
set interfaces ge-0/0/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 65000
set routing-options static route 172.16.3.16/28 discard
set routing-options static route 172.16.3.32/28 discard
```

```
set routing-options static route 172.16.3.48/28 discard
set routing-options static route 172.16.3.64/28 discard
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int neighbor 192.168.0.1 export send-static
set protocols bgp group int neighbor 192.168.0.2
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Device R4

```
set interfaces ge-0/0/0 unit 0 description to_R1
set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.6/30
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set routing-options autonomous-system 65001
set protocols bgp group ext type external
set protocols bgp group ext peer-as 65000
set protocols bgp group ext neighbor 10.1.0.5
```

Procedure

Step-by-Step Procedure

We are showing the step-by-step procedure to configure R1. The other routers have a similar step-by-step process. The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure R1:

1. Configure the interfaces.

```
[edit]
user@R1# set interfaces ge-0/0/0 unit 0 description to_R2
user@R1# set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
user@R1# set interfaces ge-0/0/1 unit 0 description to_R3
user@R1# set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.5/30
user@R1# set interfaces ge-0/0/2 unit 0 description to_R4
```

```
user@R1# set interfaces ge-0/0/2 unit 0 family inet address 10.1.0.5/30
user@R1# set interfaces lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure the internal BGP (IBGP) peerings to R2 and R3.

```
[edit]
user@R1# set protocols bgp group int type internal
user@R1# set protocols bgp group int local-address 192.168.0.1
user@R1# set protocols bgp group int neighbor 192.168.0.2
user@R1# set protocols bgp group int neighbor 192.168.0.3
```

3. Configure the external BGP (EBGP) peering to R4. The export policy configuration is shown in a later step.

```
[edit]
user@R1# set protocols bgp group to_65001 type external
user@R1# set protocols bgp group to_65001 export customer-routes
user@R1# set protocols bgp group to_65001 neighbor 10.1.0.6 peer-as 65001
```

4. Configure the OSPF peerings to R2 and R3. The OSPF protocol provides the learning of the loopback address for each device which allows the IBGP peerings to establish.

```
[edit]
user@R1# set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
user@R1# set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
user@R1# set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

5. Configure the prefix list.

```
[edit]
user@R1# set policy-options prefix-list customers 172.16.1.16/28
user@R1# set policy-options prefix-list customers 172.16.1.32/28
user@R1# set policy-options prefix-list customers 172.16.1.48/28
user@R1# set policy-options prefix-list customers 172.16.1.64/28
user@R1# set policy-options prefix-list customers 172.16.2.16/28
user@R1# set policy-options prefix-list customers 172.16.2.32/28
user@R1# set policy-options prefix-list customers 172.16.2.48/28
user@R1# set policy-options prefix-list customers 172.16.2.64/28
user@R1# set policy-options prefix-list customers 172.16.3.16/28
```

```
user@R1# set policy-options prefix-list customers 172.16.3.32/28
user@R1# set policy-options prefix-list customers 172.16.3.48/28
user@R1# set policy-options prefix-list customers 172.16.3.64/28
```

6. Configure the routing policy that references the prefix list as a match criterion.

```
[edit]
user@R1# set policy-options policy-statement customer-routes term get-routes from prefix-list
customers
user@R1# set policy-options policy-statement customer-routes term get-routes then accept
user@R1# set policy-options policy-statement customer-routes term others then reject
```

7. Configure the static routes for the 172.16.1.0/24 network. We are using static routes to emulate the customer routes.

```
[edit]
user@R1# set routing-options static route 172.16.1.16/28 discard
user@R1# set routing-options static route 172.16.1.32/28 discard
user@R1# set routing-options static route 172.16.1.48/28 discard
user@R1# set routing-options static route 172.16.1.64/28 discard
```

8. Configure the autonomous system (AS) number and router ID.

```
[edit]
user@R1# set routing-options router-id 192.168.0.1
user@R1# set routing-options autonomous-system 65000
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
ge-0/0/0 {
    unit 0 {
        description to_R2;
        family inet {
```

```
            address 10.0.0.1/30;
        }
    }
}
ge-0/0/1 {
    unit 0 {
        description to_R3;
        family inet {
            address 10.0.0.5/30;
        }
    }
}
ge-0/0/2 {
    unit 0 {
        description to_R4;
        family inet {
            address 10.1.0.5/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.1;
        neighbor 192.168.0.2;
        neighbor 192.168.0.3;
    }
    group to_65001 {
        type external;
        export customer-routes;
        neighbor 10.1.0.6 {
            peer-as 65001;
```

```
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/0.0;
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
```

```
user@R1# show policy-options
prefix-list customers {
    172.16.1.16/28;
    172.16.1.32/28;
    172.16.1.48/28;
    172.16.1.64/28;
    172.16.2.16/28;
    172.16.2.32/28;
    172.16.2.48/28;
    172.16.2.64/28;
    172.16.3.16/28;
    172.16.3.32/28;
    172.16.3.48/28;
    172.16.3.64/28;
}
policy-statement customer-routes {
    term get-routes {
        from {
            prefix-list customers;
        }
        then accept;
    }
    term others {
        then reject;
```

```
    }
}
```

```
user@R1# show routing-options
static {
    route 172.16.1.16/28 discard;
    route 172.16.1.32/28 discard;
    route 172.16.1.48/28 discard;
    route 172.16.1.64/28 discard;
}
router-id 192.168.0.1;
autonomous-system 65000;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes on R1 | 408
- Verifying the Route Advertisement to R4 | 409
- Experimenting with the prefix-list-filter Statement | 410

Confirm that the configuration is working properly.

Verifying the Routes on R1

Purpose

On R1, check the routes in the routing table.

Action

```
user@R1> show route terse 172.16/16

inet.0: 30 destinations, 30 routes (30 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
A V Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* ? 172.16.1.16/28     S   5                        Discard
* ? 172.16.1.32/28     S   5                        Discard
* ? 172.16.1.48/28     S   5                        Discard
* ? 172.16.1.64/28     S   5                        Discard
* ? 172.16.2.16/28     B 170        100                          I
  unverified                                      >10.0.0.2
* ? 172.16.2.32/28     B 170        100                          I
  unverified                                      >10.0.0.2
* ? 172.16.2.48/28     B 170        100                          I
  unverified                                      >10.0.0.2
* ? 172.16.2.64/28     B 170        100                          I
  unverified                                      >10.0.0.2
* ? 172.16.3.16/28     B 170        100                          I
  unverified                                      >10.0.0.6
* ? 172.16.3.32/28     B 170        100                          I
  unverified                                      >10.0.0.6
* ? 172.16.3.48/28     B 170        100                          I
  unverified                                      >10.0.0.6
* ? 172.16.3.64/28     B 170        100                          I
  unverified                                      >10.0.0.6
```

Meaning

Device R1 has learned its own static routes (S) and the BGP routes from Devices R2 and R3 (B).

Verifying the Route Advertisement to R4

Purpose

On R1, make sure that the customer routes are advertised to R4.

Action

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 30 destinations, 30 routes (30 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
```

```
* 172.16.1.48/28          Self                                I
* 172.16.1.64/28          Self                                I
* 172.16.2.16/28          Self                                I
* 172.16.2.32/28          Self                                I
* 172.16.2.48/28          Self                                I
* 172.16.2.64/28          Self                                I
* 172.16.3.16/28          Self                                I
* 172.16.3.32/28          Self                                I
* 172.16.3.48/28          Self                                I
* 172.16.3.64/28          Self                                I
```

Meaning

As expected, only the routes from the customer prefix list are advertised to R4.

Experimenting with the prefix-list-filter Statement

Purpose

See what can happen when you use `prefix-list-filter` instead of `prefix-list`.

Action

1. On R3, add a static route that is longer than one of the existing static routes.

```
[edit routing-options static]
user@R3# set route 172.16.3.65/32 discard
user@R3# commit
```

2. On R1, deactivate the prefix list and configure a prefix list filter with the `orlonger` match type.

```
[edit policy-options policy-statement customer-routes term get-routes]
user@R1# deactivate from prefix-list customers
user@R1# set from prefix-list-filter customers orlonger
user@R1# commit
```

3. On R1, check which routes are advertised to R4.

```
user@R1> show route advertising-protocol bgp 10.1.0.6

inet.0: 31 destinations, 31 routes (31 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.16/28          Self                                    I
* 172.16.1.32/28          Self                                    I
* 172.16.1.48/28          Self                                    I
* 172.16.1.64/28          Self                                    I
* 172.16.2.16/28          Self                                    I
* 172.16.2.32/28          Self                                    I
* 172.16.2.48/28          Self                                    I
* 172.16.2.64/28          Self                                    I
* 172.16.3.16/28          Self                                    I
* 172.16.3.32/28          Self                                    I
* 172.16.3.48/28          Self                                    I
* 172.16.3.64/28          Self                                    I
* 172.16.3.65/32          Self                                    I
```

Meaning

As expected, R1 is now advertising the 172.16.3.65/32 route to R4, even though 172.16.3.65/32 is not in the prefix list.

RELATED DOCUMENTATION

Understanding Prefix Lists for Use in Routing Policy Match Conditions | 392

Example: Configuring Policy Chains and Route Filters | 258

Example: Configuring a Policy Subroutine | 289

Example: Configuring the Priority for Route Prefixes in the RPD Infrastructure

IN THIS SECTION

- Requirements | 412
- Overview | 413
- Configuration | 414
- Verification | 421

This example shows how to configure priority for route prefixes in the RPD infrastructure for the OSPF, LDP, and BGP protocols.

Requirements

This example uses the following hardware and software components:

- Three routers in a combination of ACX Series, M Series, MX Series, PTX Series, and T Series.
- Junos OS Release 16.1 or later running on all devices.

Before you begin:

1. Configure the device interfaces.
2. Configure the following protocols:
    - BGP
    - MPLS
    - OSPF
    - LDP

Overview

IN THIS SECTION

- Topology | 413

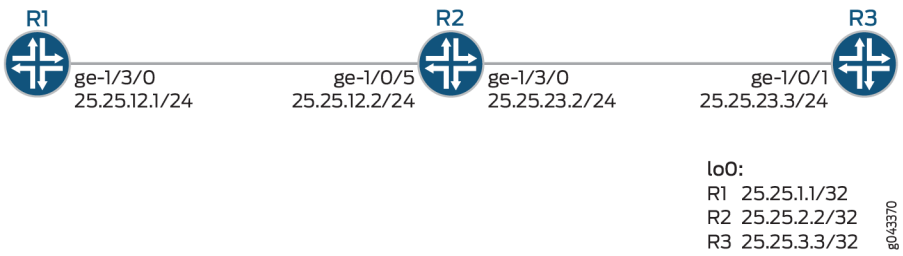
In a network with a large number of routes, it is sometimes important to control the order in which routes get updated for better convergence and to provide differentiated services. Prefix prioritization helps users to prioritize certain routes/prefixes over others and have control over the order in which routes get updated in the RIB (routing table) and the FIB (forwarding table). In Junos OS Release 16.1 and later, you can control the order in which the routes get updated from LDP/OSPF to rpd and rpd to kernel. You can specify a priority of `high` or `low` through the existing import policy in the protocols. In the event of a topology change, high priority prefixes are updated in the routing table first, followed by low priority prefixes. In general, routes that are not explicitly assigned a priority are treated as medium priority. Within the same priority level, routes will continue to be updated in lexicographic order.

In this example, the routing device is in area 0.0.0.0, with interface ge-1/3/0 connected to the neighboring device. You configure three import routing policies: next-hop-self, ospf-prio, and prio_for_bgp. The routing policy next-hop-self accepts routes from BGP. For the OSPF routing policy, routes matching 172.16.25.3/32 are installed first because they have a priority of high. LDP imports routes from OSPF. For BGP prioritization, routes matching 172.16.50.1/32 are installed first because they have a priority of high. Routes associated with these prefixes are installed in the routing table in the order of the specified priority of the prefix.

Topology

Figure 29 on page 413 shows the sample topology.

Figure 29: Priority for Route Prefixes in the rpd Infrastructure



Configuration

IN THIS SECTION

- CLI Quick Configuration | 414
- Configuring Device R1 | 416
- Results | 418

CLI Quick Configuration

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

R1

```
set interfaces ge-1/3/0 unit 0 family inet address 172.16.12.1/24
set interfaces ge-1/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.25.1/32
set protocols mpls interface ge-1/3/0.0
set protocols bgp group prio_internal type internal
set protocols bgp group prio_internal local-address 172.16.25.1
set protocols bgp group prio_internal import prio_for_bgp
set protocols bgp group prio_internal neighbor 172.16.25.3 family inet unicast
set protocols bgp group prio_internal neighbor 172.16.25.3 export next-hop-self
sset protocols ospf import ospf_prio
set protocols ospf area 0.0.0.0 interface ge-1/3/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-1/3/0.0
set protocols ldp interface lo0.0
set policy-options policy-statement next-hop-self term nhself from protocol bgp
set policy-options policy-statement next-hop-self term nhself then next-hop self
set policy-options policy-statement next-hop-self term nhself then accept
set policy-options policy-statement ospf_prio term ospf_ldp from protocol ospf
set policy-options policy-statement ospf_prio term ospf_ldp from route-filter 172.16.25.3/32
exact
set policy-options policy-statement ospf_prio term ospf_ldp then priority high
set policy-options policy-statement ospf_prio term ospf_ldp then accept
set policy-options policy-statement prio_for_bgp term bgp_prio from protocol bgp
```

```
set policy-options policy-statement prio_for_bgp term bgp_prio from route-filter 172.16.50.1/32
exact
set policy-options policy-statement prio_for_bgp term bgp_prio then priority high
set routing-options nonstop-routing
set routing-options router-id 172.16.25.1
set routing-options autonomous-system 2525
```

R2

```
set interfaces ge-1/0/5 unit 0 family inet address 172.16.12.2/24
set interfaces ge-1/0/5 unit 0 family mpls
set interfaces ge-1/3/0 unit 0 family inet address 172.16.23.2/24
set interfaces ge-1/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.25.2/32
set protocols mpls interface ge-1/0/5.0
set protocols mpls interface ge-1/3/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-1/0/5.0
set protocols ospf area 0.0.0.0 interface ge-1/3/0.0
set protocols ldp interface ge-1/0/5.0
set protocols ldp interface ge-1/3/0.0
set protocols ldp interface lo0.0
set routing-options nonstop-routing
set routing-options router-id 172.16.25.2
set routing-options autonomous-system 2525
```

R3

```
set interfaces ge-1/0/1 unit 0 family inet address 172.16.23.3/24
set interfaces ge-1/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 172.16.25.3/32
set protocols mpls interface ge-1/0/1.0
set protocols bgp group prio_internal type internal
set protocols bgp group prio_internal local-address 172.16.25.3
set protocols bgp group prio_internal neighbor 172.16.25.1 family inet unicast
set protocols bgp group prio_internal neighbor 172.16.25.1 export next-hop-self
set protocols bgp group prio_internal neighbor 172.16.25.1 export static_to_bgp
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-1/0/1.0
set protocols ldp interface ge-1/0/1.0
set protocols ldp interface lo0.0
```

```
set policy-options policy-statement next-hop-self term nhself from protocol bgp
set policy-options policy-statement next-hop-self term nhself then next-hop self
set policy-options policy-statement next-hop-self term nhself then accept
set policy-options policy-statement static_to_bgp term s_to_b from protocol static
set policy-options policy-statement static_to_bgp term s_to_b from route-filter 172.16.50.1/32
exact
set policy-options policy-statement static_to_bgp term s_to_b from route-filter 172.16.50.2/32
exact
set policy-options policy-statement static_to_bgp term s_to_b then accept
set routing-options nonstop-routing
set routing-options static route 172.16.50.1/32 receive
set routing-options static route 172.16.50.2/32 receive
set routing-options router-id 172.16.25.3
set routing-options autonomous-system 2525
```

Configuring Device R1

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the CLI User Guide.

To configure Device R1:

1. Configure the interfaces.

```
[edit interfaces]
user@R1# set interfaces ge-1/3/0 unit 0 family inet address 172.16.12.1/24
user@R1# set interfaces ge-1/3/0 unit 0 family mpls
user@R1# set interfaces lo0 unit 0 family inet address 172.16.25.1/32
```

2. Assign the loopback address to the device.

```
[edit lo0 unit 0 family]
user@R1# set address 172.16.25.1/32
```

3. Configure MPLS.

```
[edit protocols]
user@R1# set protocols mpls interface ge-1/3/0.0
```

4. Configure the router ID and autonomous system of Router R1.

```
[edit routing-options]
user@R1# set router-id 172.16.7.7
user@R1# set autonomous-system 100
```

5. Enable OSPF on the interfaces of Router R1.

```
[edit protocols]
user@R1# set protocols ospf import ospf_prio
user@R1# set protocols ospf area 0.0.0.0 interface ge-1/3/0.0
user@R1# set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

6. Configure LDP protocols on the interfaces.

```
[edit protocols]
user@R1# set protocols ldp interface ge-1/3/0.0
user@R1# set protocols ldp interface lo0.0
```

7. Configure BGP.

```
[edit protocols]
user@R1# set protocols bgp group prio_internal type internal
user@R1# set protocols bgp group prio_internal local-address 172.16.25.1
user@R1# set protocols bgp group prio_internal import prio_for_bgp
user@R1# set protocols bgp group prio_internal neighbor 172.16.25.3 family inet unicast
user@R1# set protocols bgp group prio_internal neighbor 172.16.25.3 export next-hop-self
```

8. Configure the policy options to prioritize the routes. The policy next-hop-self accepts routes from BGP. You configure three import routing policies: next-hop-self, ospf-prio, and prio_for_bgp. The routing policy next-hop-self accepts routes from BGP. For the ospf-prio routing policy, routes matching 172.16.25.3/32 are installed first because they have a priority of high. LDP imports routes

from OSPF. For prio_for_bgp policy, routes matching 172.16.50.1/32 are installed first because they have a priority of high.

```
[edit policy-options policy-statement]
user@R1# set policy-options policy-statement next-hop-self term nhself from protocol bgp
user@R1# set policy-options policy-statement next-hop-self term nhself then next-hop self
user@R1# set policy-options policy-statement next-hop-self term nhself then accept
user@R1# set policy-options policy-statement ospf_prio term ospf_ldp from protocol ospf
user@R1# set policy-options policy-statement ospf_prio term ospf_ldp from route-filter
172.16.25.3/32 exact
 set policy-options policy-statement ospf_prio term ospf_ldp then priority high
 set policy-options policy-statement ospf_prio term ospf_ldp then accept
 set policy-options policy-statement prio_for_bgp term bgp_prio from protocol bgp
 set policy-options policy-statement prio_for_bgp term bgp_prio from route-filter
172.16.50.1/32 exact
 set policy-options policy-statement prio_for_bgp term bgp_prio then priority high
```

Results

From configuration mode, confirm your configuration by entering the **show interfaces**, **show protocols**, **show routing-options**, and **show policy-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R1# show interfaces
ge-1/3/0 {
    unit 0 {
        family inet {
            address 172.16.12.1/24;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address address 172.16.25.1/32;
        }
```

```
    }
}
```

```
[edit]
user@R1# show protocols
mpls {
    interface ge-1/3/0.0;
}
bgp {
    group prio_internal {
        type internal;
        local-address 172.16.25.1;
    import prio_for_bgp
        neighbor 172.16.25.3 {
            family inet {
                unicast;
                }
                export next-hop-self;
                }
                }
            }
        ospf {
            import ospf_prio;
            area 0.0.0.0 {
                interface ge-1/3/0.0;
                interface lo0.0 {
                    passive;
                    }
            }
        }
        ldp {
        interface ge-1/3/0.0;
        interface lo0.0;
        }
    }
```

```
[edit]
user@R1# show routing-options
nonstop-routing;
```

```
router-id 172.16.25.1;
autonomous-system 2525;
```

```
[edit]
user@R1# show policy-options
policy-statement next-hop-self {
    term nhself {
        from protocol bgp;
        then {
            next-hop self;
                accept;
            }
    }
}
policy-statement ospf_prio {
    term ospf_ldp {
        from {
            protocol ospf;
            route-filter 172.16.25.3/32 exact;
            }
        then {
                priority high;
                accept;
                }
            }
    }
policy-statement prio_for_bgp {
    term bgp_prio {
        from {
            protocol bgp;
        route-filter 172.16.50.1/32 exact;
            }
            then priority high;
    }
}
```

If you are done configuring the device, enter `commit` from the configuration mode.

Verification

IN THIS SECTION

- Verifying the Priority for OSPF Routes | 421
- Verifying the Priority for LDP Routes | 422
- Verifying the Priority for BGP Routes | 424

Confirm that the configuration is working properly.

Verifying the Priority for OSPF Routes

Purpose

Verify that the priority is set for the expected route in OSPF.

Action

On Device R1, from operational mode, run the `show ospf route 172.16.25.3/32 extensive` command. A priority of high is applied to OSPF route 172.16.25.3.

```
user@R1> show ospf route 172.16.25.3/32 extensive

Topology default Route Table:

Prefix             Path  Route      NH       Metric NextHop       Nexthop
                   Type  Type       Type            Interface     Address/LSP
172.16.25.3          Intra Router     IP            2 ge-1/3/0.0    172.16.12.2
  area 0.0.0.0, origin 172.16.25.3, optional-capability 0x0
172.16.25.3/32       Intra Network    IP            2 ge-1/3/0.0    172.16.12.2
  area 0.0.0.0, origin 172.16.25.3, priority high
```

Meaning

The output shows priority `high` is applied for OSPF route 172.16.25.3.

Verifying the Priority for LDP Routes

Purpose

Verify if LDP inherits from OSPF.

Action

From operational mode, enter the `show route 172.16.25.3` command to verify if LDP has inherited routes from OSPF.

```
user@R1> show route 172.16.25.3

  inet.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.25.3/32       *[OSPF/10] 00:10:27, metric 2
                    > to 172.16.25.2 via ge-1/3/0.0

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.25.3/32       *[LDP/9] 00:10:24, metric 1
                    > to 172.16.25.2 via ge-1/3/0.0, Push 299824
```

From operational mode, enter the `show route 172.16.25.3 extensive` command to verify if LDP has inherited priority.

```
user@R1> show route 172.16.25.3 extensive
inet.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
172.16.25.3/32 (1 entry, 1 announced)
        State:<Flashall>
TSI:
KRT in-kernel 172.16.25.3/32 -> {172.16.12.2}
        *OSPF   Preference: 10
                Next hop type: Router, Next hop index: 549
                Address: 0xa463390
                Next-hop reference count: 6
                Next hop: 172.16.12.2 via ge-1/3/0.0, selected
                Session Id: 0x0
```

```
                State:<Active Int HighPriority>
                Local AS:  2525
                Age: 10:43      Metric: 2
                Validation State: unverified
                Area: 0.0.0.0
                Task: OSPF
                Announcement bits (4): 0-KRT 4-LDP 6-Resolve tree 2 7-Resolve_IGP_FRR task
                AS path: I


inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

172.16.25.3/32 (1 entry, 1 announced)
        State:<Flashall>
LDP     Preference: 9
                Next hop type: Router, Next hop index: 582
                Address: 0xa477810
                Next-hop reference count: 12
                Next hop: 172.16.12.2 via ge-1/3/0.0, selected
                Label operation: Push 299824
                Label TTL action: prop-ttl
                Load balance label: Label 299824: None;
                Label element ptr: 0xa17ad00
                Label parent element ptr: 0x0
                Label element references: 1
                Label element child references: 0
                Label element lsp id: 0
                Session Id: 0x0
                State:<Active Int HighPriority>
                Local AS:  2525
                Age: 10:40      Metric: 1
                Validation State: unverified
                Task: LDP
                Announcement bits (3): 2-Resolve tree 1 3-Resolve tree 2 4-Resolve_IGP_FRR task
                AS path: I
```

Meaning

The output shows that LDP inherits priority `high` for route 172.16.25.3 from OSPF.

Verifying the Priority for BGP Routes

Purpose

Verify that priority is set for the expected route in BGP.

Action

On Device R1, from operational mode, run the `show route protocol bgp` command to display the routes learned from BGP.

```
user@R1> show route protocol bgp

inet.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.50.1/32        *[BGP/170] 00:11:24, localpref 100, from 172.16.25.3
                       AS path: I, validation-state: unverified
                     > to 172.16.12.2 via ge-1/3/0.0, Push 299824
172.16.50.2/32        *[BGP/170] 00:11:24, localpref 100, from 172.16.25.3
                       AS path: I, validation-state: unverified
                     > to 172.16.12.2 via ge-1/3/0.0, Push 299824

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

mpls.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
```

On Device R1, from operational mode, run the `show route 172.16.50.1 extensive` command. High priority is applied for BGP route 172.16.50.1.

```
user@R1> show route 172.16.50.1 extensive

inet.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
172.16.50.1/32 (1 entry, 1 announced)
TSI:
KRT in-kernel 172.16.50.1/32 -> {indirect(1048574)}
        *BGP    Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0xa487b10
                Next-hop reference count: 4
                Source: 172.16.25.3
```

```
                Next hop type: Router, Next hop index: 582
                Next hop: 172.16.12.2 via ge-1/3/0.0, selected
                Label operation: Push 299824
                Label TTL action: prop-ttl
                Load balance label: Label 299824: None;
                Label element ptr: 0xa17ad00
                Label parent element ptr: 0x0
                Label element references: 1
                Label element child references: 0
                Label element lsp id: 0
                Session Id: 0x0
                Protocol next hop: 172.16.25.3
                Indirect next hop: 0xa4a9800 1048574 INH Session ID: 0x0
                State: <Active Int Ext HighPriority>
                Local AS:  2525 Peer AS:  2525
                Age: 11:49      Metric2: 1
                Validation State: unverified
                Task: BGP_2525.172.16.25.3
                Announcement bits (2): 0-KRT 6-Resolve tree 2
                AS path: I (Atomic)
                Accepted
                Localpref: 100
                Router ID: 172.16.25.3
                Indirect next hops: 1
                        Protocol next hop: 172.16.25.3 Metric: 1
                        Indirect next hop: 0xa4a9800 1048574 INH Session ID: 0x0
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 172.16.12.2 via ge-1/3/0.0
                                Session Id: 0x0
                        172.16.25.3/32 Originating RIB: inet.3
                          Metric: 1                       Node path count: 1
                          Forwarding nexthops: 1
                                Nexthop: 172.16.12.2 via ge-1/3/0.0
```

Meaning

The output shows that priority `high` is applied for BGP route 172.16.50.1.

RELATED DOCUMENTATION

Prefix Prioritization Overview | 15

Configuring Priority for Route Prefixes in RPD Infrastructure | 426

Configuring Priority for Route Prefixes in RPD Infrastructure

Prefix prioritization helps users to prioritize certain routes or prefixes for better convergence and to provide differentiated services. In a network with a large number of routes, it is sometimes important to control the order in which routes get updated due to changes in the network topology. At a system level, Junos OS implements reasonable defaults based on heuristics to determine the order in which routes get updated. However, the default behavior is not always optimal. Prefix prioritization provides the user the ability to control the order in which the routes get updated from LDP or OSPF to rpd, and rpd to kernel. The Junos OS policy language is extended to let the user set relative priority (high and low) for prefixes through the existing import policy in protocols. Based on the tagged priority, the routes are placed in different priority queues. In the event of a topology change, high priority prefixes are updated in the routing table first, followed by low priority prefixes. Within the same priority level, routes will continue to be updated in lexicographic order. Routes that are not explicitly assigned a priority are treated as medium priority.

Before you begin to configure prefix prioritization in rpd for protocols such as OSPF, LDP, and BGP:

- Configure the router interfaces.
- Configure MPLS.
- Configure the OSPF, BGP, and LDP protocols.

To configure the priority `high` for the OSPF protocol:

1. Configure the policy term.

```
[edit policy-options policy-statement policy-name]
user@host# set term term-name
```

For example:

```
[edit policy-options policy-statement ospf-prio]
user@host# set term t1
```

2. Configure the policy term to accept routes from OSPF.

```
[edit policy-options policy-statement ospf-prio  term t1]
user@host# set from protocol ospf
```

3. Specify the desired route as a match condition for which you want to set priority `high`.

```
[edit policy-options policy-statement ospf-prio term t1]
user@host# set from route-filter destination-prefix match-type
```

For example:

```
[edit policy-options policy-statement ospf-prio term t1]
user@host# set from route-filter 172.16.25.3/32 exact
```

4. Specify that the route is to be accepted and set priority `high` for the route if the previous conditions are matched.

```
[edit policy-options policy-statement ospf-prio term t1]
user@host# set then priority high
user@host# set then accept
```

5. Verify the configuration.

```
[edit]
user@host# show policy-options
policy-statement ospf-prio {
    term t1 {
        from {
            route-filter 172.16.25.3/32 exact;
            }
        then {
                priority high;
                accept;
                }
            }
    }
```

LDP inherits from OSPF.

To configure priority `high` for LDP:

1. Configure the policy term that imports from OSPF.

```
[edit policy-options policy-statement policy-name]
user@host# set term term-name
```

For example:

```
[edit policy-options policy-statement ospf-import]
user@host# set term ospf_ldp
```

2. Configure the term to accept routes and priority from OSPF.

```
[edit policy-options policy-statement ospf_import term ospf_ldp]
user@host# set from protocol ospf
user@host# set from route-filter destination-prefix match-type
```

For example:

```
[edit policy-options policy-statement ospf_import term ospf_ldp]
user@host# set from protocol ospf
user@host# set from route-filter 172.16.25.3/32 exact
```

3. Verify the configuration.

```
[edit]
user@host# show policy-options
policy-statement ospf-import {
    term ospf_ldp {
        from {
            protocol ospf ;
            route-filter 172.16.25.3/32 exact;
            }
        then {
                priority high;
                accept;
                }
```

```
            }
    }
```

To configure the priority `high` for the BGP protocol:

1. Configure the policy term.

   ```
   [edit policy-options policy-statement policy-name]
   user@host# set term term-name
   ```

   For example:

   ```
   [edit policy-options policy-statement prio-for-bgp]
   user@host# set term bgp_prio
   ```

2. Specify the desired route as a match condition.

   ```
   [edit policy-options policy-statement prio-for-bgp term bgp_prio]
   user@host# set from protocol bgp
   user@host# set from route-filter destination-prefix match-type
   ```

   For example:

   ```
   [edit policy-options policy-statement prio-for-bgp term bgp_prio]
   user@host# set from protocol bgp
   user@host# set from route-filter 172.16.50.1/32 exact
   ```

3. Specify that the route is to be accepted and set the priority `high` for the route if the previous conditions are matched.

   ```
   [edit policy-options policy-statement prio-for-bgp term bgp_prio]
   user@host# set then priority high
   user@host# set then accept
   ```

4. Verify the configuration.

   ```
   policy-statement prio_for_bgp {
       term bgp_prio {
   ```

```
        from {
            protocol bgp;
            route-filter 172.16.50.1/32 exact;
            }
                then {
                    priority high;
                    accept;
                    }
    }
}
```

NOTE: For BGP, you can also configure priority based on the route-distinguisher (RD) value in case of L3VPN. For example, you can configure priority for BGP with route-distinguisher 51.51.51.51:111.

To configure priority for BGP based on the route-distinguisher (RD) value:

1. Configure the policy term.

```
[edit policy-options policy-statement policy-name]
user@host# set term term-name
```

For example:

```
[edit policy-options policy-statement prio-for-bgp]
user@host# set term bgp_prio
```

2. Specify the desired route as a match condition.

```
[edit policy-options policy-statement prio-for-bgp term bgp_prio]
user@host# set from rib bgp.l3vpn.0
user@host# set from route-filter destination-prefix match-type
user@host# set from route-distinguisher route-distinguisher value
```

For example:

```
[edit policy-options policy-statement prio-for-bgp term bgp_prio]
user@host# set from rib bgp.l3vpn.0
```

```
user@host# set from route-filter 172.16.1.1/32 exact
user@host# set from route-distinguisher RD1
```

3. Specify that the route is to be accepted and set the priority `high` for the route if the previous conditions are matched.

```
[edit policy-options policy-statement prio-for-bgp term bgp_prio]
user@host# set then priority high
user@host# set then accept
```

4. Verify the configuration.

```
policy-statement prio_for_bgp {
    term bgp_prio {
        from {
            protocol rib bgp.l3vpn.0;
            route-filter 172.16.1.1/32 exact;
            route-distinguisher RD1;
            }
                then {
                    priority high;
                    accept;
                    }
    }
}
```

NOTE: Low priority prefixes are installed only after the high priority prefixes in the routing table. You can also configure priority `low` similarly to priority `high` for the routes you want to set to low priority.

NOTE: Priority is applied only when routes are pushed from RIB to FIB. Therefore, you cannot modify the priority of routes that are already installed. Changing the priority of routes already installed does not make sense. If you try changing the priority of routes already installed, there is a show output difference:

```
user@R1> show route 172.16.25.3 extensive | match state
 State: <FlashAll>
```

```
              State:   <Active Int HighPriority>     <=== OSPF
        Validation State: unverified
        State: <FlashAll>
              State:   <Active Int>                  <=== LDP
        Validation State: unverified
```

As the route is already installed in FIB, LDP does not show the priority as High.

Restarting the routing daemon to remove the routes and adding it again reflects the proper priority from both the OSPF and LDP protocol perspective.

```
user@R1> restart routing
Routing protocols process signalled but still running, waiting 8 seconds more
Routing protocols process started, pid 4512

user@R1> show route 172.16.25.3 extensive |match state
 State: <FlashAll>
              State:   <Active Int HighPriority>     <=== OSPF
        Validation State: unverified
        State: <FlashAll>
              State:   <Active Int HighPriority>     <=== LDP
        Validation State: unverified
```

RELATED DOCUMENTATION

Prefix Prioritization Overview | 15

Example: Configuring the Priority for Route Prefixes in the RPD Infrastructure | 412

CHAPTER 5

Configuring AS Paths as Match Conditions

IN THIS CHAPTER

- Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions | **433**
- Example: Using AS Path Regular Expressions | **442**
- Understanding Prepending AS Numbers to BGP AS Paths | **457**
- Example: Configuring a Routing Policy for AS Path Prepending | **458**
- Understanding Adding AS Numbers to BGP AS Paths | **468**
- Example: Advertising Multiple Paths in BGP | **469**
- Improve the Performance of AS Path Lookup in BGP Policy | **506**

Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions

IN THIS SECTION

- Configuration of AS Path Regular Expressions | **434**
- How AS Path Regular Expressions Are Evaluated | **440**
- Examples: Configuring AS Path Regular Expressions | **441**

A BGP *AS path* is the sequence of autonomous systems that network packets traverse to get to a specified router. AS numbers are assembled in a sequence 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 5 and so on until it reaches AS 1. In this case, AS 1 is the last AS before the packet destination; it is the AS that the source of the packet would peer with.

When working with AS paths and routing policy match conditions, you can use regular expressions to locate routes. To do so, create one or more match conditions based on some or all of the AS path, and then include it in a routing policy.

The following sections describe AS path regular expressions and provide configuration examples.

Configuration of 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 "Routing Policy Match Conditions" on page 58). To create a named AS path regular expression, include the `as-path` statement:

```
[edit policy-options]
as-path name regular-expression;
```

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

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.

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

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.

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

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

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

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

```
term <operator>
```

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

    In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. You can configure a value in the range from 1 through 4,294,967,295.
- *operator*—(Optional) An operator specifying how the term must match. Most operators describe how many times the term must be found to be considered a match (for example, any number of occurrences, or zero, or one occurrence). Table 19 on page 436 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 20 on page 437 shows examples of how to define regular expressions to match AS paths.

Table 19: AS Path Regular Expression Operators

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

Table 20: Examples of AS Path Regular Expressions

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

Table 20: Examples of AS Path Regular Expressions *(Continued)*

AS Path to Match	Regular Expression	Sample Matches
	[123–125]*	Null AS path 123 124 124 125 125 125 123 124 125 123
Path whose second AS number must be 56 or 78	(. 56) | (. 78) or . (56 | 78)	1234 56 1234 78 9876 56 3857 78
Path whose second AS number might be 56 or 78	. (56 | 78)?	1234 56 52 34 56 1234 1234 78 39 794 78 2
Path whose first AS number is 123 and second AS number is either 56 or 78	123 (56|78)	123 56 123 78
Path of any length, except nonexistent, whose second AS number can be anything, including nonexistent	. .* or . .{0,}	1234 1234 5678 1234 5 6 7 8
AS path is 1 2 3	1 2 3	1 2 3
One occurrence of the AS numbers 1 and 2, followed by one or more occurrences of the AS number 3	1 2 3+	1 2 3 1 2 3 3 1 2 3 3 3

Table 20: Examples of AS Path Regular Expressions *(Continued)*

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

Configuring a Null AS Path

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

```
[edit policy-options]
as-path null-as “()";
```

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]
as-path 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 {
    as-path 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 {
    as-path 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 {
```

```
    as-path addison "123-125";
    policy-statement from-addison {
        from as-path addison;
        }
        then {
            preference 200;
            accept;
        }
    }
}
```

RELATED DOCUMENTATION

Example: Using AS Path Regular Expressions | 442

Example: Configuring a Routing Policy for AS Path Prepending | 458

Example: Using AS Path Regular Expressions

IN THIS SECTION

- Requirements | 442
- Overview | 443
- Configuration | 445
- Verification | 453

An autonomous system (AS) path is a route attribute used by BGP. The AS path is used both for route selection and to prevent potential routing loops. This example shows how to use regular expressions with AS path numbers to locate a set of routes.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

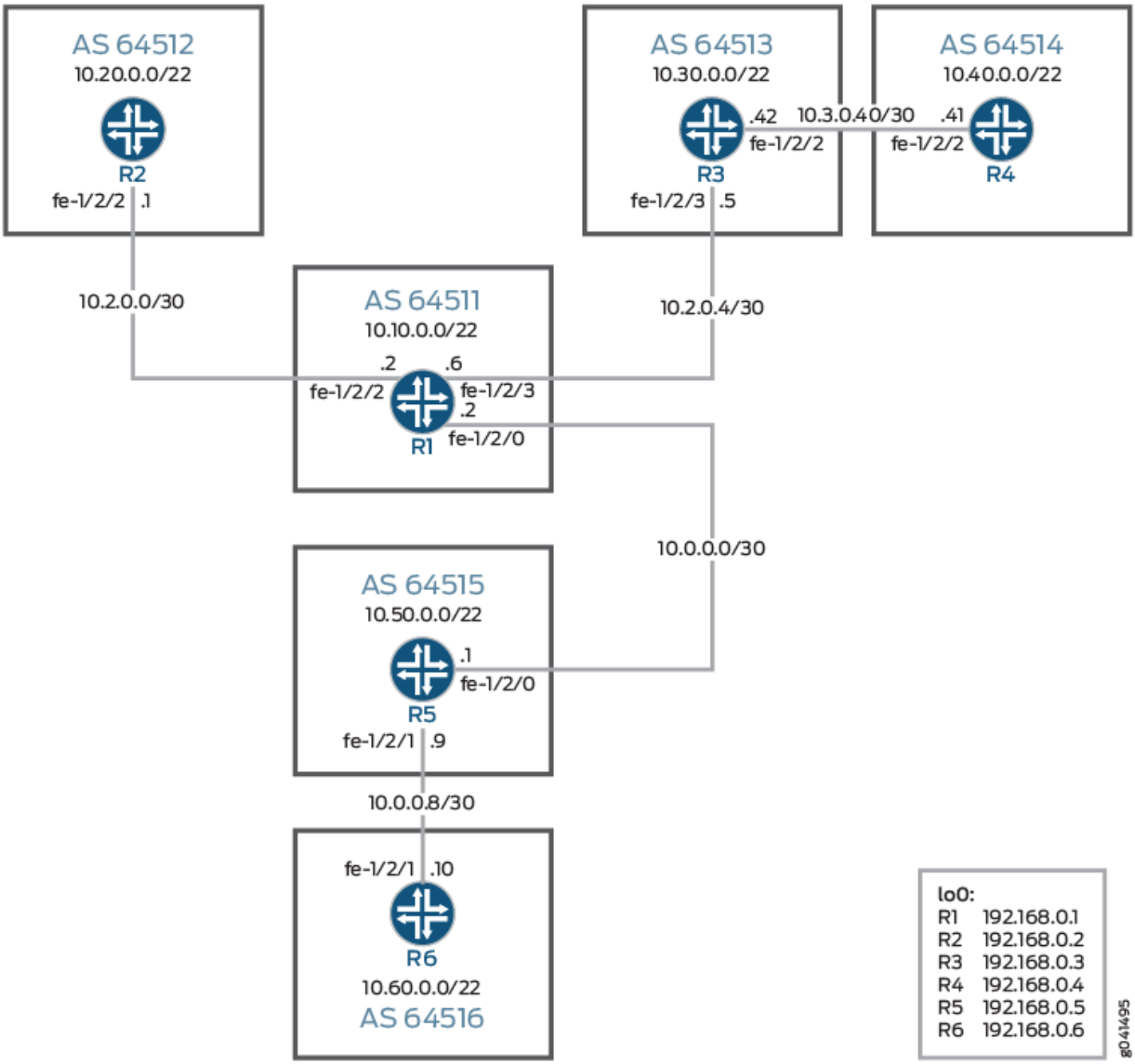
- Topology | **443**

Figure 30 on page 444 shows several ASs connected through external BGP (EBGP) peering sessions. Each device is generating customer routes within its assigned address space.

Topology

Figure 30 on page 444 shows the sample network.

Figure 30: BGP Topology AS Regular Expressions



The administrators of AS 64516 want to reject all routes originating in AS 64513 and AS 64514. Two AS path regular expressions called `orig-in-64513` and `orig-in-64514` are created and referenced in a policy called `reject-some-routes`. The routing policy is then applied as an import policy on Device R6.

"CLI Quick Configuration" on page 445 shows the configuration for all of the devices in Figure 30 on page 444.

The section "No Link Title" on page 448 describes the steps on Device R2 and Device R6. "Verification" on page 453 shows how to use the `aspath-regex` option with the `show route` command on Device R2 to locate routes using regular expressions.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 445
- Procedure | 448

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/2 unit 0 description to-R2
set interfaces fe-1/2/2 unit 0 family inet address 10.2.0.2/30
set interfaces fe-1/2/3 unit 0 description to-R3
set interfaces fe-1/2/3 unit 0 family inet address 10.2.0.6/30
set interfaces fe-1/2/0 unit 0 description to-R5
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp export send-static
set protocols bgp group 64512 type external
set protocols bgp group 64512 peer-as 64512
set protocols bgp group 64512 neighbor 10.2.0.1
set protocols bgp group 64513 type external
set protocols bgp group 64513 peer-as 64513
set protocols bgp group 64513 neighbor 10.2.0.5
set protocols bgp group 64515 type external
set protocols bgp group 64515 peer-as 64515
set protocols bgp group 64515 neighbor 10.0.0.1
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.10.1.0/24 reject
set routing-options static route 10.10.2.0/24 reject
set routing-options static route 10.10.3.0/24 reject
set routing-options autonomous-system 64511
```

Device R2

```
set interfaces fe-1/2/2 unit 0 description to-R1
set interfaces fe-1/2/2 unit 0 family inet address 10.2.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp export send-static
set protocols bgp group 64511 type external
set protocols bgp group 64511 peer-as 64511
set protocols bgp group 64511 neighbor 10.2.0.2
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.20.1.0/24 reject
set routing-options static route 10.20.2.0/24 reject
set routing-options static route 10.20.3.0/24 reject
set routing-options autonomous-system 64512
```

Device R3

```
set interfaces fe-1/2/3 unit 0 description to-R1
set interfaces fe-1/2/3 unit 0 family inet address 10.2.0.5/30
set interfaces fe-1/2/2 unit 0 description to-R4
set interfaces fe-1/2/2 unit 0 family inet address 10.3.0.42/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp export send-static
set protocols bgp group 64511 type external
set protocols bgp group 64511 peer-as 64511
set protocols bgp group 64511 neighbor 10.2.0.6
set protocols bgp group 64514 type external
set protocols bgp group 64514 peer-as 64514
set protocols bgp group 64514 neighbor 10.3.0.41
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.30.1.0/24 reject
set routing-options static route 10.30.2.0/24 reject
set routing-options static route 10.30.3.0/24 reject
set routing-options autonomous-system 64513
```

Device R4

```
set interfaces fe-1/2/2 unit 0 description to-R3
set interfaces fe-1/2/2 unit 0 family inet address 10.3.0.41/30
```

```
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set protocols bgp export send-static
set protocols bgp group 64513 type external
set protocols bgp group 64513 peer-as 64513
set protocols bgp group 64513 neighbor 10.3.0.42
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.40.1.0/24 reject
set routing-options static route 10.40.2.0/24 reject
set routing-options static route 10.40.3.0/24 reject
set routing-options autonomous-system 64514
```

Device R5

```
set interfaces fe-1/2/0 unit 0 description to-R1
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/1 unit 0 description to-R6
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.9/30
set interfaces lo0 unit 0 family inet address 192.168.0.5/32
set protocols bgp export send-static
set protocols bgp group 64511 type external
set protocols bgp group 64511 peer-as 64511
set protocols bgp group 64511 neighbor 10.0.0.2
set protocols bgp group 64516 type external
set protocols bgp group 64516 peer-as 64516
set protocols bgp group 64516 neighbor 10.0.0.10
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.50.1.0/24 reject
set routing-options static route 10.50.2.0/24 reject
set routing-options static route 10.50.3.0/24 reject
set routing-options autonomous-system 64515
```

Device R6

```
set interfaces fe-1/2/1 unit 0 description to-R5
set interfaces fe-1/2/1 unit 0 family inet address 10.0.0.10/30
set interfaces lo0 unit 0 family inet address 192.168.0.6/32
set protocols bgp export send-static
set protocols bgp group 64515 type external
set protocols bgp group 64515 import reject-some-routes
```

```
set protocols bgp group 64515 peer-as 64515
set protocols bgp group 64515 neighbor 10.0.0.9
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set policy-options policy-statement reject-some-routes term find-routes from as-path orig-
in-64513
set policy-options policy-statement reject-some-routes term find-routes from as-path orig-
in-64514
set policy-options policy-statement reject-some-routes term find-routes then reject
set policy-options as-path orig-in-64513 ".* 64513"
set policy-options as-path orig-in-64514 ".* 64514"
set routing-options static route 10.60.1.0/24 reject
set routing-options static route 10.60.2.0/24 reject
set routing-options static route 10.60.3.0/24 reject
set routing-options autonomous-system 64516
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/2 unit 0 description to-R1
user@R2# set fe-1/2/2 unit 0 family inet address 10.2.0.1/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the EBGP connection to Device R1.

```
[edit protocols bgp]
user@R2# set export send-static
user@R2# set group 64511 type external
user@R2# set group 64511 peer-as 64511
user@R2# set group 64511 neighbor 10.2.0.2
```

3. Configure the routing policy.

```
[edit policy-options policy-statement send-static term 1]
user@R2# set from protocol static
user@R2# set then accept
```

4. Configure the static routes.

```
[edit routing-options static]
user@R2# set route 10.20.1.0/24 reject
user@R2# set route 10.20.2.0/24 reject
user@R2# set route 10.20.3.0/24 reject
```

5. Configure the AS number.

```
[edit routing-options]
user@R2# set autonomous-system 64512
```

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R6:

1. Configure the device interfaces.

```
[edit interfaces]
user@R6# set fe-1/2/1 unit 0 description to-R5
user@R6# set fe-1/2/1 unit 0 family inet address 10.0.0.10/30
user@R6# set lo0 unit 0 family inet address 192.168.0.6/32
```

2. Configure the EBGP connection to Device R5.

```
[edit protocols bgp]
user@R6# set export send-static
user@R6# set group 64515 type external
```

```
user@R6# set group 64515 import reject-some-routes
user@R6# set group 64515 peer-as 64515
user@R6# set group 64515 neighbor 10.0.0.9
```

3. Configure the routing policy that sends static routes.

```
[edit policy-options policy-statement send-static term 1]
user@R6# set  from protocol static
user@R6# set then accept
```

4. Configure the routing policy that rejects certain routes.

```
[edit policy-options policy-statement reject-some-routes term find-routes]
user@R6# set from as-path orig-in-64513
user@R6# set from as-path orig-in-64514
user@R6# set then reject
[edit policy-options]
user@R6# set as-path orig-in-64513 ".* 64513"
user@R6# set as-path orig-in-64514 ".* 64514"
```

5. Configure the static routes.

```
[edit routing-options static]
user@R6# set route 10.60.1.0/24 reject
user@R6# set route 10.60.2.0/24 reject
user@R6# set route 10.60.3.0/24 reject
```

6. Configure the AS number.

```
[edit routing-options]
user@R6# set autonomous-system 64516
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R2

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        description to-R1;
        family inet {
            address 10.2.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    export send-static;
    group 64511 {
        type external;
        peer-as 64511;
        neighbor 10.2.0.2;
    }
}
```

```
user@R2# show policy-options
policy-statement send-static {
    term 1 {
        from protocol static;
        then accept;
```

```
    }
}
```

```
user@R2# show routing-options
static {
    route 10.20.1.0/24 reject;
    route 10.20.2.0/24 reject;
    route 10.20.3.0/24 reject;
}
autonomous-system 64512;
```

Device R6

```
user@R6# show interfaces
fe-1/2/0 {
    unit 0 {
        description to-R5;
        family inet {
            address 10.0.0.10/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.6/32;
        }
    }
}
```

```
user@R6# show protocols
bgp {
    export send-static;
    group 64515 {
        type external;
        import reject-some-routes;
        peer-as 64515;
        neighbor 10.0.0.9;
```

```
    }
}
```

```
user@R6# show policy-options
policy-statement reject-some-routes {
    term find-routes {
        from as-path [ orig-in-64513 orig-in-64514 ];
        then reject;
    }
}
policy-statement send-static {
    term 1 {
        from protocol static;
        then accept;
    }
}
as-path orig-in-64513 ".* 64513";
as-path orig-in-64514 ".* 64514";
```

```
user@R6# show routing-options
static {
    route 10.60.1.0/24 reject;
    route 10.60.2.0/24 reject;
    route 10.60.3.0/24 reject;
}
autonomous-system 64516;
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Finding Routes on Device R2 | **454**
- Making Sure That Routes Are Excluded on Device R6 | **455**

Confirm that the configuration is working properly.

Finding Routes on Device R2

Purpose

On Device R2, use the `show route aspath-regex` command to locate routes using regular expressions.

Action

Look for routes that are originated by Device R6 in AS 64516.

```
user@R2> show route terse aspath-regex ".* 64516"

inet.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A V Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* ? 10.60.1.0/24       B 170        100                           64511 64515 64516 I
  unverified                                      >10.2.0.2
* ? 10.60.2.0/24       B 170        100                           64511 64515 64516 I
  unverified                                      >10.2.0.2
* ? 10.60.3.0/24       B 170        100                           64511 64515 64516 I
  unverified                                      >10.2.0.2
```

Look for routes that are originated in either AS 64514 or AS 64516.

```
user@R2> show route terse aspath-regex ".* (64514|64516)"

inet.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A V Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* ? 10.40.1.0/24       B 170        100                           64511 64513 64514 I
  unverified                                      >10.2.0.2
* ? 10.40.2.0/24       B 170        100                           64511 64513 64514 I
  unverified                                      >10.2.0.2
* ? 10.40.3.0/24       B 170        100                           64511 64513 64514 I
  unverified                                      >10.2.0.2
* ? 10.60.1.0/24       B 170        100                           64511 64515 64516 I
  unverified                                      >10.2.0.2
* ? 10.60.2.0/24       B 170        100                           64511 64515 64516 I
  unverified                                      >10.2.0.2
```

```
* ? 10.60.3.0/24       B 170        100                          64511 64515 64516 I
  unverified                                    >10.2.0.2
```

Look for routes that use AS 64513 as a transit network.

```
user@R2> show route terse aspath-regex ".* 64513 .+"

inet.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A V Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* ? 10.40.1.0/24       B 170        100                          64511 64513 64514 I
  unverified                                    >10.2.0.2
* ? 10.40.2.0/24       B 170        100                          64511 64513 64514 I
  unverified                                    >10.2.0.2
* ? 10.40.3.0/24       B 170        100                          64511 64513 64514 I
  unverified
```

Meaning

The output shows the routing table entries that match the specified AS path regular expressions.

Making Sure That Routes Are Excluded on Device R6

Purpose

On Device R6, use the `show route` and `show route hidden` commands to make sure that routes originating from AS 64513 and AS 64514 are excluded from Device R6's routing table.

Action

```
user@R6> show route 10.30.0/22
inet.0: 21 destinations, 21 routes (15 active, 0 holddown, 6 hidden)
```

```
user@R6> show route 10.40.0/22
inet.0: 21 destinations, 21 routes (15 active, 0 holddown, 6 hidden)
```

```
user@R6> show route hidden

inet.0: 21 destinations, 21 routes (15 active, 0 holddown, 6 hidden)
+ = Active Route, - = Last Active, * = Both

10.30.1.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
10.30.2.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
10.30.3.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
10.40.1.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 64514 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
10.40.2.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 64514 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
10.40.3.0/24        [BGP ] 02:24:47, localpref 100
                      AS path: 64515 64511 64513 64514 I, validation-state: unverified
                    > to 10.0.0.9 via fe-1/2/1.0
```

Meaning

The output shows that the 10.30.0/22 and 10.40.0/22 routes are rejected on Device R6.

RELATED DOCUMENTATION

Understanding AS Path Regular Expressions for Use as Routing Policy Match Conditions | 433

Example: Testing a Routing Policy with Complex Regular Expressions | 760

Understanding 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.

The BGP best path algorithm determines how the best path to an autonomous system (AS) is selected. The AS path length determines the best path when all of the following conditions are met:

- There are multiple potential routes to an AS.
- BGP has the lowest preference value (sometimes referred to as administrative distance) of the available routes.
- The local preferences of the available routes are equal.

When these conditions are met, the AS path length is used as the tie breaker in the best path algorithm. When two or more routes exist to reach a particular prefix, BGP prefers the route with the shortest AS Path length.

If you are an enterprise that has multihoming to one or more service providers, you might prefer that incoming traffic take a particular path to reach your network. Perhaps you have two connections, but one costs less than the other. Or you might have one fast connection and another, much slower connection that you only want to use as a backup if your primary connection is down. AS path prepending is an easy method that you can use to influence inbound routing to your AS.

In Junos OS Release 9.1 and later, you can specify 4-byte AS numbers as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*, as well as the 2-byte AS numbers that are supported in earlier releases of the Junos OS. In plain-number format, you can configure a value in the range from 1 through 4,294,967,295.

If you have a router that does not support 4-byte AS numbers in the AS path, the prependend AS number displayed in the AS path is the AS_TRANS number, AS 23456. To display the route details, use the *show route* command.

RELATED DOCUMENTATION

Example: Configuring a Routing Policy for AS Path Prepending | 458

Example: Using AS Path Regular Expressions | 442

Understanding BGP Path Selection

Example: Configuring a Routing Policy for AS Path Prepending

IN THIS SECTION

- Requirements | 458
- Overview | 458
- Configuration | 459
- Verification | 464
- Appendix Full Configurations | 467

This example shows how to configure a routing policy to prepend the AS path on specific routes advertised by BGP.

Requirements

Before you begin, make sure your router interfaces and protocols are correctly configured. We provide the interface and BGP protocol configuration used in this document.

NOTE: This example was updated and re-validated on Junos release 22.1R1.

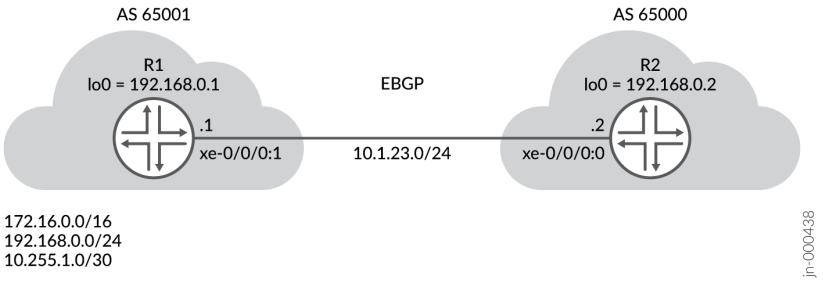
Overview

IN THIS SECTION

- Topology | 459

In this example, you create a routing policy called *prependpolicy1* and a term called *prependterm1*. The routing policy prepends AS number 65001 three times to routes that match the 172.16.0.0/12, 192.168.0.0/16, and 10.0.0.0/8 prefixes, when the mask length is equal to or longer than the specified mask. The result is a match occurs when the route's mask length is equal to or longer than the specified network mask. The *prependpolicy1* policy is applied as an export policy to the BGP routes advertised by R1 in AS 65001 to R2 in AS number 65000. Routes that don't match the specified prefix ranges do not undergo AS path prepending.

Topology



In the topology EBGP peering is configured between R1 and R2. Direct interface peering to the 10.1.23.0/24 subnet addresses is used. R1 belongs to AS number 65001 and is configured to prepend its AS number to a specific set of matching routes when advertised to R2.

By adding AS numbers to the AS path the route becomes less likely to be selected for forwarding. This might be done by the owner of AS 65001 to reduce the amount of ingress traffic it receives from the operator of AS 65000.

NOTE: In this example we demonstrate AS path prepending through an export policy. You can also use an import policy to match on routes for attribute manipulation. In general its a best practice to only prepend your local AS number to routes. Prepending AS numbers that belong to remote networks can lead to unexpected results.

For details on BGP paths selection see *Understanding BGP Path Selection*.

Configuration

IN THIS SECTION

- Procedure | 460

Procedure

CLI Quick Configuration

In this section we focus on the configuration of the R1 device. Refer to the appendix for the complete configurations of all devices used in this example.

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

In this example we assign three test prefixes to an unused interface on R1. A fourth test prefix is assigned to R1's loopback address. This provides four direct routes that can be advertised into BGP. Our policy uses a combination of `protocol direct` and `route-filter` statements to control which prefixes undergo AS path prepending.

```
set system host-name R1

set interfaces xe-0/0/0:1 unit 0 family inet address 10.1.23.1/24
set interfaces xe-0/0/0:0 unit 0 family inet address 10.255.1.1/30
set interfaces xe-0/0/0:0 unit 0 family inet address 172.16.0.1/24
set interfaces xe-0/0/0:0 unit 0 family inet address 10.200.1.1/24
set interfaces lo0 unit 0 family inet address 192.168.0.1/32

set policy-options policy-statement prependpolicy1 term prependterm1 from protocol direct
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
172.16.0.0/16 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
192.168.0.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
10.255.1.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 then as-path-prepend "65001
65001 65001"
set policy-options policy-statement prependpolicy1 term prependterm1 then accept
set policy-options policy-statement prependpolicy1 term else from protocol direct
set policy-options policy-statement prependpolicy1 term else from route-filter 10.200.0.0/16
orlonger
set policy-options policy-statement prependpolicy1 term else then accept

set routing-options autonomous-system 65001
set routing-options router-id 192.168.0.1

set protocols bgp group ebgp type external
```

```
set protocols bgp group ebgp export prependpolicy1
set protocols bgp group ebgp peer-as 65000
set protocols bgp group ebgp neighbor 10.1.23.2
```

Step-by-Step Procedure

The following steps requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To create a routing policy that prepends AS numbers to specific routes:

1. Configure the peering and loopback interfaces.

```
user@R1#
[edit]
set interfaces xe-0/0/0:1 unit 0 family inet address 10.1.23.1/24
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure the AS number, RID, and the external BGP peer group. You define the *prependpolicy1* policy in the next step. The policy is applied as an export policy to affect the routes advertised by R1.

```
user@R1#
[edit]
set routing-options autonomous-system 65001
set routing-options router-id 192.168.0.1

set protocols bgp group ebgp type external
set protocols bgp group ebgp export prependpolicy1
set protocols bgp group ebgp peer-as 65000
set protocols bgp group ebgp neighbor 10.1.23.2
```

3. Configure the *prependpolicy1* policy. The use of `or-longer` switch to the route filter statements allows a match when the mask length is equal to or longer than the specified mask. Other options like `exact` match only when the prefix and mask lengths are equal. The *else* term demonstrates how a route

that does not match the *prependterm1* term is advertised without AS path prepending by matching the *else* term.

```
user@R1#
[edit]
set policy-options policy-statement prependpolicy1 term prependterm1 from protocol direct
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
172.16.0.0/16 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
192.168.0.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
10.255.1.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 then as-path-prepend
"65001 65001 65001"
set policy-options policy-statement prependpolicy1 term prependterm1 then accept
set policy-options policy-statement prependpolicy1 term else from protocol direct
set policy-options policy-statement prependpolicy1 term else from route-filter 10.200.0.0/16
orlonger
set policy-options policy-statement prependpolicy1 term else then accept
```

NOTE: When you enter multiple AS numbers, you must separate each number with a space. Enclose the string of AS numbers in double quotation marks.

4. Define test routes. In our sample topology we assign prefixes to an unused interface that is operationally up. This provides direct routes for BGP to advertise for testing the operation of the export policy.

```
user@R1#
[edit]
set interfaces xe-0/0/0:0 unit 0 family inet address 10.255.1.1/30
set interfaces xe-0/0/0:0 unit 0 family inet address 172.16.0.1/24
set interfaces xe-0/0/0:0 unit 0 family inet address 10.200.1.1/24
```

Results

Confirm your configuration by entering the **show policy-options**, **show protocols bgp**, **show routing-options**, and **show interfaces** commands from configuration mode. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@R1#[edit]
user@R1# show policy-options
policy-statement prependpolicy1 {
    term prependterm1 {
        from {
            protocol direct;
            route-filter 172.16.0.0/16 orlonger;
            route-filter 192.168.0.0/24 orlonger;
            route-filter 10.255.1.0/24 orlonger;
        }
        then {
            as-path-prepend "65001 65001 65001";
            accept;
        }
    }
    term else {
        from {
            protocol direct;
            route-filter 10.200.0.0/16 orlonger;
        }
        then accept;
    }
}

[edit]
user@R1# show protocols bgp
group ebgp {
    type external;
    export direct;
    peer-as 65000;
    neighbor 10.1.23.2;
}

[edit]
user@R1# show routing-options
autonomous-system 65001;
```

```
router-id 192.168.0.1

user@R1# show interfaces xe-0/0/0:0
unit 0 {
    family inet {
        address 10.255.1.1/30;
        address 172.16.0.1/24;
        address 10.200.1.1/24;
    }
}
[edit]
user@R1# show interfaces xe-0/0/0:1
unit 0 {
    family inet {
        address 10.1.23.1/24;
    }
}
[edit]
user@R1# show interfaces lo0
unit 0 {
    family inet {
        address 192.168.0.1/32;
    }
}
```

If you are done configuring the R1 device, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying the AS Prepending Policy | **465**
- Verifying Routing Policy Application and BGP Peering | **465**
- Verify AS Path Prepending | **466**

To confirm that the configuration is working properly, perform these tasks:

Verifying the AS Prepending Policy

Purpose

Verify that the policy is configured on the device, and that the appropriate routes are specified to prepend with AS numbers.

Action

From operational mode, enter the **show policy *prependpolicy1*** command.

```
user@R1> show policy prependpolicy1
Policy prependpolicy1:  [CHANGED/RESOLVED/]
    Term prependterm1:
        from proto Direct
             route filter:
                 172.16.0.0/16 orlonger
                 192.168.0.0/24 orlonger
                 10.255.1.0/24 orlonger
        then aspathprepend 65001 65001 65001 accept
    Term else:
        from proto Direct
             route filter:
                 10.200.0.0/16 orlonger
        then accept
```

The policy displays the correct match conditions and actions.

Verifying Routing Policy Application and BGP Peering

Purpose

Verify the routing policy is applied as an export policy to the EBGP peer group. This step also confirms the BGP session to R2 is correctly established.

Action

From operational mode, enter the **show bgp neighbor 10.1.23.2** command.

```
user@R1> show bgp neighbor 10.1.23.2
Peer: 10.1.23.2+49642 AS 65000 Local: 10.1.23.1+179 AS 65001
```

```
  Group: ebgp                  Routing-Instance: master
  Forwarding routing-instance: master
  Type: External    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Export: [ prependpolicy1 ]
  Options: <PeerAS Refresh>
  Options: <GracefulShutdownRcv>
  Holdtime: 90 Preference: 170
  Graceful Shutdown Receiver local-preference: 0
  Number of flaps: 1
  Last flap event: RecvNotify
  Error: 'Cease' Sent: 0 Recv: 1
  Peer ID: 192.168.0.2   Local ID: 192.168.0.1    Active Holdtime: 90
  . . .
  Input messages:  Total 2498   Updates 1       Refreshes 0     Octets 47510
  Output messages: Total 2500   Updates 3       Refreshes 0     Octets 47620
  Output Queue[1]: 0            (inet.0, inet-unicast)
```

The command output confirms the BGP session is established and that R1 has applied the *prependpolicy1* policy as export.

Verify AS Path Prepending

Purpose

Verify the export policy works as design to prepend AS numbers to matching routes.

Action

From operational mode, enter the **show route protocol bgp** command on R2. Alternatively, use the **show route advertising-protocol bgp 10.1.23.2** at R1 to display details about the routes it advertises to R2.

```
user@R2> show route protocol bgp

inet.0: 14 destinations, 14 routes (14 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.1.0/24      *[BGP/170] 00:04:46, localpref 100
                      AS path: 65001 I, validation-state: unverified
                    >  to 10.1.23.1 via xe-0/0/0:0.0
10.255.1.0/30      *[BGP/170] 00:04:46, localpref 100
```

```
                    AS path: 65001 65001 65001 65001 I, validation-state: unverified
                  >  to 10.1.23.1 via xe-0/0/0:0.0
172.16.0.0/24     *[BGP/170] 00:04:46, localpref 100
                    AS path: 65001 65001 65001 65001 I, validation-state: unverified
                  >  to 10.1.23.1 via xe-0/0/0:0.0
192.168.0.1/32    *[BGP/170] 00:04:46, localpref 100
                    AS path: 65001 65001 65001 65001 I, validation-state: unverified
                  >  to 10.1.23.1 via xe-0/0/0:0.0
```

The routes show the expected AS path prepending. Note that the 10.200.1.0/24 route only has one instance of AS number 65001. This route does not match the route filter statements in the *prependterm1* of the *prependpolicy1* policy and so does not undergo any prepending.

R1's view of the BGP routes it advertises to R2 is provided for completeness:

```
user@R1> show route advertising-protocol bgp 10.1.23.2

inet.0: 16 destinations, 16 routes (16 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.200.1.0/24           Self                                    I
* 10.255.1.0/30           Self                                    65001 65001 65001 [65001] I
* 172.16.0.0/24           Self                                    65001 65001 65001 [65001] I
* 192.168.0.1/32          Self                                    65001 65001 65001 [65001] I
```

Appendix Full Configurations

The full configuration for R1.

```
set system host-name R1
set interfaces xe-0/0/0:0 unit 0 family inet address 10.255.1.1/30
set interfaces xe-0/0/0:0 unit 0 family inet address 172.16.0.1/24
set interfaces xe-0/0/0:0 unit 0 family inet address 10.200.1.1/24
set interfaces xe-0/0/0:1 unit 0 family inet address 10.1.23.1/24
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set policy-options policy-statement prependpolicy1 term prependterm1 from protocol direct
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
172.16.0.0/16 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
192.168.0.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 from route-filter
10.255.1.0/24 orlonger
set policy-options policy-statement prependpolicy1 term prependterm1 then as-path-prepend "65001
```

```
65001 65001"
set policy-options policy-statement prependpolicy1 term prependterm1 then accept
set policy-options policy-statement prependpolicy1 term else from protocol direct
set policy-options policy-statement prependpolicy1 term else from route-filter 10.200.0.0/16
orlonger
set policy-options policy-statement prependpolicy1 term else then accept
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 65001
set protocols bgp group ebgp type external
set protocols bgp group ebgp export prependpolicy1
set protocols bgp group ebgp peer-as 65000
set protocols bgp group ebgp neighbor 10.1.23.2
```

The full configuration for R2.

```
set system host-name R2
set interfaces xe-0/0/0:0 unit 0 family inet address 10.1.23.2/24
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 65000
set protocols bgp group ebgp type external
set protocols bgp group ebgp peer-as 65001
set protocols bgp group ebgp neighbor 10.1.23.1
```

Understanding 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.

NOTE: If you are configuring both `as-path-expand` and `as-path-prepend` policy actions in a routing policy, ensure that you configure `as-path-expand` before configuring `as-path-prepend` to avoid the misplacement of the AS numbers, which can result in an incorrect AS path calculation.

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.

RELATED DOCUMENTATION

Example: Advertising Multiple Paths in BGP | 469

Example: Configuring a Routing Policy for AS Path Prepending | 458

Example: Advertising Multiple Paths in BGP

IN THIS SECTION

- Requirements | 470
- Overview | 470
- Configuration | 472
- Verification | 499

In this example, BGP routers are configured to advertise multiple paths instead of advertising only the active path. Advertising multiple paths in BGP is specified in RFC 7911, *Advertisement of Multiple Paths in BGP*.

Requirements

This example uses the following hardware and software components:

- Eight BGP-enabled devices.
- Five of the BGP-enabled devices do not necessarily need to be routers. For example, they can be EX Series Ethernet Switches.
- Three of the BGP-enabled devices are configured to send multiple paths or receive multiple paths (or both send and receive multiple paths). These three BGP-enabled devices must be M Series Multiservice Edge Routers, MX Series 5G Universal Routing Platforms, or T Series Core Routers.
- The three routers must be running Junos OS Release 11.4 or later.

Overview

IN THIS SECTION

- Topology Diagram | 471

The following statements are used for configuring multiple paths to a destination:

```
[edit protocols bgp group group-name family family]
add-path {
    receive;
    send {
        include-backup-path include-backup-path;
        multipath;
        path-count path-count;
        path-selection-mode {
            (all-paths | equal-cost-paths);
        }
        prefix-policy [ policy-names ... ];
    }
}
```

In this example, Router R5, Router R6, and Router R7 redistribute static routes into BGP. Router R1 and Router R4 are route reflectors. Router R2 and Router R3 are clients to Route Reflector R1. Router R8 is a client to Route Reflector R4.

Route reflection is optional when multiple-path advertisement is enabled in BGP.

With the `add-path send path-count 6` configuration, Router R1 is configured to send up to six paths (per destination) to Router R4.

With the `add-path receive` configuration, Router R4 is configured to receive multiple paths from Router R1.

With the `add-path send path-count 6` configuration, Router R4 is configured to send up to six paths to Router R8.

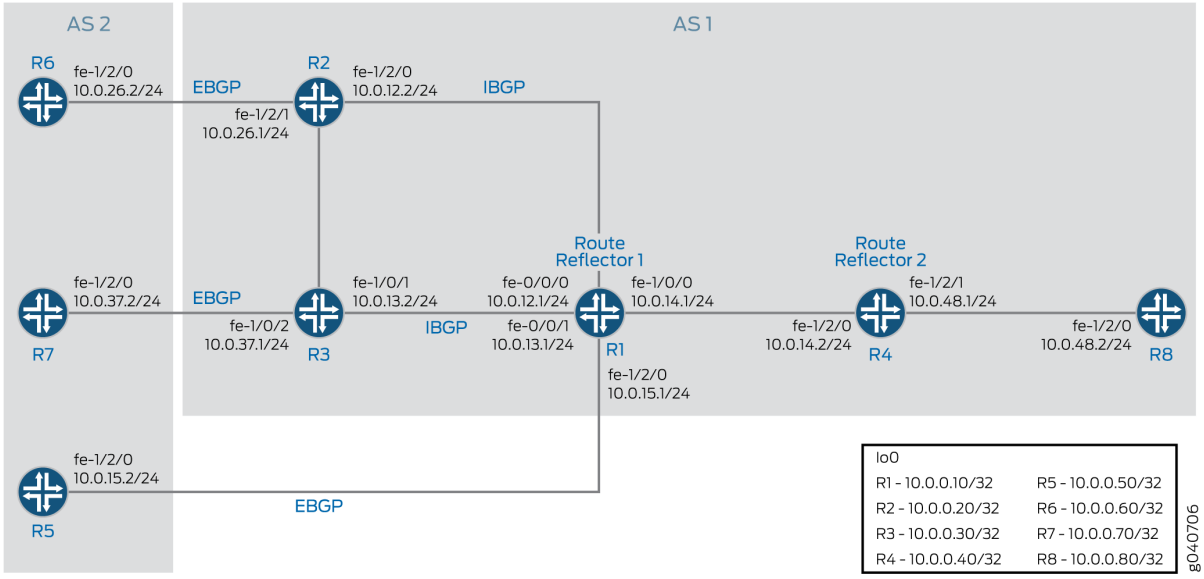
With the `add-path receive` configuration, Router R8 is configured to receive multiple paths from Router R4.

The `add-path send prefix-policy allow_199` policy configuration (along with the corresponding route filter) limits Router R4 to sending multiple paths for only the 172.16.199.1/32 route.

Topology Diagram

Figure 31 on page 471 shows the topology used in this example.

Figure 31: Advertisement of Multiple Paths in BGP



Configuration

IN THIS SECTION

- CLI Quick Configuration | 472
- Configuring Router R1 | 476
- Configuring Router R2 | 479
- Configuring Router R3 | 482
- Configuring Router R4 | 485
- Configuring Router R5 | 489
- Configuring Router R6 | 492
- Configuring Router R7 | 494
- Configuring Router R8 | 496
- Results | 498

CLI Quick Configuration

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

Router R1

```
set interfaces fe-0/0/0 unit 12 family inet address 10.0.12.1/24
set interfaces fe-0/0/1 unit 13 family inet address 10.0.13.1/24
set interfaces fe-1/0/0 unit 14 family inet address 10.0.14.1/24
set interfaces fe-1/2/0 unit 15 family inet address 10.0.15.1/24
set interfaces lo0 unit 10 family inet address 10.0.0.10/32
set protocols bgp group rr type internal
set protocols bgp group rr local-address 10.0.0.10
set protocols bgp group rr cluster 10.0.0.10
set protocols bgp group rr neighbor 10.0.0.20
set protocols bgp group rr neighbor 10.0.0.30
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.15.2 local-address 10.0.15.1
set protocols bgp group e1 neighbor 10.0.15.2 peer-as 2
set protocols bgp group rr_rr type internal
```

```
set protocols bgp group rr_rr local-address 10.0.0.10
set protocols bgp group rr_rr neighbor 10.0.0.40 family inet unicast add-path send path-count 6
set protocols ospf area 0.0.0.0 interface lo0.10 passive
set protocols ospf area 0.0.0.0 interface fe-0/0/0.12
set protocols ospf area 0.0.0.0 interface fe-0/0/1.13
set protocols ospf area 0.0.0.0 interface fe-1/0/0.14
set protocols ospf area 0.0.0.0 interface fe-1/2/0.15
set routing-options router-id 10.0.0.10
set routing-options autonomous-system 1
```

Router R2

```
set interfaces fe-1/2/0 unit 21 family inet address 10.0.12.2/24
set interfaces fe-1/2/1 unit 26 family inet address 10.0.26.1/24
set interfaces lo0 unit 20 family inet address 10.0.0.20/32
set protocols bgp group rr type internal
set protocols bgp group rr local-address 10.0.0.20
set protocols bgp group rr neighbor 10.0.0.10 export set_nh_self
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.26.2 peer-as 2
set protocols ospf area 0.0.0.0 interface lo0.20 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.21
set protocols ospf area 0.0.0.0 interface fe-1/2/1.28
set policy-options policy-statement set_nh_self then next-hop self
set routing-options autonomous-system 1
```

Router R3

```
set interfaces fe-1/0/1 unit 31 family inet address 10.0.13.2/24
set interfaces fe-1/0/2 unit 37 family inet address 10.0.37.1/24
set interfaces lo0 unit 30 family inet address 10.0.0.30/32
set protocols bgp group rr type internal
set protocols bgp group rr local-address 10.0.0.30
set protocols bgp group rr neighbor 10.0.0.10 export set_nh_self
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.37.2 peer-as 2
set protocols ospf area 0.0.0.0 interface lo0.30 passive
set protocols ospf area 0.0.0.0 interface fe-1/0/1.31
set protocols ospf area 0.0.0.0 interface fe-1/0/2.37
set policy-options policy-statement set_nh_self then next-hop self
set routing-options autonomous-system 1
```

Router R4

```
set interfaces fe-1/2/0 unit 41 family inet address 10.0.14.2/24
set interfaces fe-1/2/1 unit 48 family inet address 10.0.48.1/24
set interfaces lo0 unit 40 family inet address 10.0.0.40/32
set protocols bgp group rr type internal
set protocols bgp group rr local-address 10.0.0.40
set protocols bgp group rr family inet unicast add-path receive
set protocols bgp group rr neighbor 10.0.0.10
set protocols bgp group rr_client type internal
set protocols bgp group rr_client local-address 10.0.0.40
set protocols bgp group rr_client cluster 10.0.0.40
set protocols bgp group rr_client neighbor 10.0.0.80 family inet unicast add-path send path-
count 6
set protocols bgp group rr_client neighbor 10.0.0.80 family inet unicast add-path send prefix-
policy allow_199
set protocols ospf area 0.0.0.0 interface fe-1/2/0.41
set protocols ospf area 0.0.0.0 interface lo0.40 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/1.48
set policy-options policy-statement allow_199 from route-filter 172.16.199.1/32 exact
set policy-options policy-statement allow_199 term match_199 from prefix-list match_199
set policy-options policy-statement allow_199 then add-path send-count 20
set policy-options policy-statement allow_199 then accept
set routing-options autonomous-system 1
```

Router R5

```
set interfaces fe-1/2/0 unit 51 family inet address 10.0.15.2/24
set interfaces lo0 unit 50 family inet address 10.0.0.50/32
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.15.1 export s2b
set protocols bgp group e1 neighbor 10.0.15.1 peer-as 1
set policy-options policy-statement s2b from protocol static
set policy-options policy-statement s2b from protocol direct
set policy-options policy-statement s2b then as-path-expand 2
set policy-options policy-statement s2b then accept
set routing-options autonomous-system 2
set routing-options static route 172.16.199.1/32 reject
set routing-options static route 172.16.198.1/32 reject
```

Router R6

```
set interfaces fe-1/2/0 unit 62 family inet address 10.0.26.2/24
set interfaces lo0 unit 60 family inet address 10.0.0.60/32
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.26.1 export s2b
set protocols bgp group e1 neighbor 10.0.26.1 peer-as 1
set policy-options policy-statement s2b from protocol static
set policy-options policy-statement s2b from protocol direct
set policy-options policy-statement s2b then accept
set routing-options autonomous-system 2
set routing-options static route 172.16.199.1/32 reject
set routing-options static route 172.16.198.1/32 reject
```

Router R7

```
set interfaces fe-1/2/0 unit 73 family inet address 10.0.37.2/24
set interfaces lo0 unit 70 family inet address 10.0.0.70/32
set protocols bgp group e1 type external
set protocols bgp group e1 neighbor 10.0.37.1 export s2b
set protocols bgp group e1 neighbor 10.0.37.1 peer-as 1
set policy-options policy-statement s2b from protocol static
set policy-options policy-statement s2b from protocol direct
set policy-options policy-statement s2b then accept
set routing-options autonomous-system 2
set routing-options static route 172.16.199.1/32 reject
```

Router R8

```
set interfaces fe-1/2/0 unit 84 family inet address 10.0.48.2/24
set interfaces lo0 unit 80 family inet address 10.0.0.80/32
set protocols bgp group rr type internal
set protocols bgp group rr local-address 10.0.0.80
set protocols bgp group rr neighbor 10.0.0.40 family inet unicast add-path receive
set protocols ospf area 0.0.0.0 interface lo0.80 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.84
set routing-options autonomous-system 1
```

Configuring Router R1

Step-by-Step Procedure

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

To configure Router R1:

1. Configure the interfaces to Router R2, Router R3, Router R4, and Router R5, and configure the loopback (lo0) interface.

```
[edit interfaces]
user@R1# set fe-0/0/0 unit 12 family inet address 10.0.12.1/24
user@R1# set fe-0/0/1 unit 13 family inet address 10.0.13.1/24
user@R1# set fe-1/0/0 unit 14 family inet address 10.0.14.1/24
user@R1# set fe-1/2/0 unit 15 family inet address 10.0.15.1/24
user@R1#set lo0 unit 10 family inet address 10.0.0.10/32
```

2. Configure BGP on the interfaces, and configure IBGP route reflection.

```
[edit protocols bgp]
user@R1# set group rr type internal
user@R1# set group rr local-address 10.0.0.10
user@R1# set group rr cluster 10.0.0.10
user@R1# set group rr neighbor 10.0.0.20
user@R1# set group rr neighbor 10.0.0.30
user@R1# set group rr_rr type internal
user@R1# set group rr_rr local-address 10.0.0.10
user@R1# set group e1 type external
user@R1# set group e1 neighbor 10.0.15.2 local-address 10.0.15.1
user@R1# set group e1 neighbor 10.0.15.2 peer-as 2
```

3. Configure Router R1 to send up to six paths to its neighbor, Router R4.

   The destination of the paths can be any destination that Router R1 can reach through multiple paths.

```
[edit protocols bgp]
user@R1# set group rr_rr neighbor 10.0.0.40 family inet unicast add-path send path-count 6
```

4. Configure OSPF on the interfaces.

```
[edit protocols ospf]
user@R1# set area 0.0.0.0 interface lo0.10 passive
user@R1# set area 0.0.0.0 interface fe-0/0/0.12
user@R1# set area 0.0.0.0 interface fe-0/0/1.13
user@R1# set area 0.0.0.0 interface fe-1/0/0.14
user@R1# set area 0.0.0.0 interface fe-1/2/0.15
```

5. Configure the router ID and the autonomous system number.

```
[edit routing-options]
user@R1# set router-id 10.0.0.10
user@R1# set autonomous-system 1
```

6. If you are done configuring the device, commit the configuration.

```
user@R1# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
fe-0/0/0 {
    unit 12 {
        family inet {
            address 10.0.12.1/24;
        }
    }
}
fe-0/0/1 {
    unit 13 {
        family inet {
            address 10.0.13.1/24;
        }
```

```
    }
}
fe-1/0/0 {
    unit 14 {
        family inet {
            address 10.0.14.1/24;
        }
    }
}
fe-1/2/0 {
    unit 15 {
        family inet {
            address 10.0.15.1/24;
        }
    }
}
lo0 {
    unit 10 {
        family inet {
            address 10.0.0.10/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group rr {
        type internal;
        local-address 10.0.0.10;
        cluster 10.0.0.10;
        neighbor 10.0.0.20;
        neighbor 10.0.0.30;
    }
    group e1 {
        type external;
        neighbor 10.0.15.2 {
            local-address 10.0.15.1;
            peer-as 2;
        }
    }
    group rr_rr {
```

```
        type internal;
        local-address 10.0.0.10;
        neighbor 10.0.0.40 {
            family inet {
                unicast {
                    add-path {
                        send {
                            path-count 6;
                        }
                    }
                }
            }
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.10 {
            passive;
        }
        interface fe-0/0/0.12;
        interface fe-0/0/1.13;
        interface fe-1/0/0.14;
        interface fe-1/2/0.15;
    }
}
```

```
user@R1# show routing-options
router-id 10.0.0.10;
 autonomous-system 1;
```

Configuring Router R2

Step-by-Step Procedure

To configure Router R2:

1. Configure the loopback (lo0) interface and the interfaces to Router R6 and Router R1.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 21 family inet address 10.0.12.2/24
user@R2# set fe-1/2/1 unit 26 family inet address 10.0.26.1/24
user@R2# set lo0 unit 20 family inet address 10.0.0.20/32
```

2. Configure BGP and OSPF on Router R2's interfaces.

```
[edit protocols]
user@R2# set bgp group rr type internal
user@R2# set bgp group rr local-address 10.0.0.20
user@R2# set bgp group e1 type external
user@R2# set bgp group e1 neighbor 10.0.26.2 peer-as 2
user@R2# set ospf area 0.0.0.0 interface lo0.20 passive
user@R2# set ospf area 0.0.0.0 interface fe-1/2/0.21
user@R2# set ospf area 0.0.0.0 interface fe-1/2/1.28
```

3. For routes sent from Router R2 to Router R1, advertise Router R2 as the next hop, because Router R1 does not have a route to Router R6's address on the 10.0.26.0/24 network.

```
[edit]
user@R2# set policy-options policy-statement set_nh_self then next-hop self
user@R2# set protocols bgp group rr neighbor 10.0.0.10 export set_nh_self
```

4. Configure the autonomous system number.

```
[edit]
user@R2# set routing-options autonomous-system 1
```

5. If you are done configuring the device, commit the configuration.

```
user@R2# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`,and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 21 {
        family inet {
            address 10.0.12.2/24;
        }
    }
}
fe-1/2/1 {
    unit 26 {
        family inet {
            address 10.0.26.1/24;
        }
    }
}
lo0 {
    unit 20 {
        family inet {
            address 10.0.0.20/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group rr {
        type internal;
        local-address 10.0.0.20;
        neighbor 10.0.0.10 {
            export set_nh_self;
        }
    }
    group e1 {
        type external;
        neighbor 10.0.26.2 {
```

```
            peer-as 2;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.20 {
            passive;
        }
        interface fe-1/2/0.21;
        interface fe-1/2/1.28;
    }
}
```

```
user@R2# show policy-options
policy-statement set_nh_self {
    then {
        next-hop self;
    }
}
```

```
user@R2# show routing-options
 autonomous-system 1;
```

Configuring Router R3

Step-by-Step Procedure

To configure Router R3:

1. Configure the loopback (lo0) interface and the interfaces to Router R7 and Router R1.

```
[edit interfaces]
user@R3# set fe-1/0/1 unit 31 family inet address 10.0.13.2/24
user@R3# set fe-1/0/2 unit 37 family inet address 10.0.37.1/24
user@R3# set lo0 unit 30 family inet address 10.0.0.30/32
```

2. Configure BGP and OSPF on Router R3's interfaces.

```
[edit protocols]
user@R3# set bgp group rr type internal
user@R3# set bgp group rr local-address 10.0.0.30
user@R3# set bgp group e1 type external
user@R3# set bgp group e1 neighbor 10.0.37.2 peer-as 2
user@R3# set ospf area 0.0.0.0 interface lo0.30 passive
user@R3# set ospf area 0.0.0.0 interface fe-1/0/1.31
user@R3# set ospf area 0.0.0.0 interface fe-1/0/2.37
```

3. For routes sent from Router R3 to Router R1, advertise Router R3 as the next hop, because Router R1 does not have a route to Router R7's address on the 10.0.37.0/24 network.

```
[edit]
user@R3# set policy-options policy-statement set_nh_self then next-hop self
user@R3# set protocols bgp group rr neighbor 10.0.0.10 export set_nh_self
```

4. Configure the autonomous system number.

```
[edit]
user@R3# set routing-options autonomous-system 1
```

5. If you are done configuring the device, commit the configuration.

```
user@R3# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
fe-1/0/1 {
    unit 31 {
        family inet {
```

```
            address 10.0.13.2/24;
        }
    }
}
fe-1/0/2 {
    unit 37 {
        family inet {
            address 10.0.37.1/24;
        }
    }
}
lo0 {
    unit 30 {
        family inet {
            address 10.0.0.30/32;
        }
    }
}
```

```
user@R3# show protocols
bgp {
    group rr {
        type internal;
        local-address 10.0.0.30;
        neighbor 10.0.0.10 {
            export set_nh_self;
        }
    }
    group e1 {
        type external;
        neighbor 10.0.37.2 {
            peer-as 2;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.30 {
            passive;
        }
        interface fe-1/0/1.31;
```

```
        interface fe-1/0/2.37;
    }
}
user@R3# show policy-options
policy-statement set_nh_self {
    then {
        next-hop self;
    }
}
```

```
user@R3# show routing-options
 autonomous-system 1;
```

Configuring Router R4

Step-by-Step Procedure

To configure Router R4:

1. Configure the interfaces to Router R1 and Router R8, and configure the loopback (lo0) interface.

```
[edit interfaces]
user@R4# set fe-1/2/0 unit 41 family inet address 10.0.14.2/24
user@R4# set fe-1/2/1 unit 48 family inet address 10.0.48.1/24
user@R4# set lo0 unit 40 family inet address 10.0.0.40/32
```

2. Configure BGP on the interfaces, and configure IBGP route reflection.

```
[edit protocols bgp]
user@R4# set group rr type internal
user@R4# set group rr local-address 10.0.0.40
user@R4# set group rr neighbor 10.0.0.10
user@R4# set group rr_client type internal
user@R4# set group rr_client local-address 10.0.0.40
user@R4# set group rr_client cluster 10.0.0.40
```

3. Configure Router R4 to send up to six paths to its neighbor, Router R8.

The destination of the paths can be any destination that Router R4 can reach through multiple paths.

```
[edit protocols bgp]
user@R4# set group rr_client neighbor 10.0.0.80 family inet unicast add-path send path-count 6
```

4. Configure Router R4 to receive multiple paths from its neighbor, Router R1.

   The destination of the paths can be any destination that Router R1 can reach through multiple paths.

```
[edit protocols bgp group rr family inet unicast]
user@R4# set add-path receive
```

5. Configure OSPF on the interfaces.

```
[edit protocols ospf area 0.0.0.0]
user@R4# set interface fe-1/2/0.41
user@R4# set interface lo0.40 passive
user@R4# set interface fe-1/2/1.48
```

6. Configure a policy that allows Router R4 to send Router R8 multiple paths to the 172.16.199.1/32 route.

   - Router R4 receives multiple paths for the 172.16.198.1/32 route and the 172.16.199.1/32 route. However, because of this policy, Router R4 only sends multiple paths for the 172.16.199.1/32 route.

```
[edit protocols bgp group rr_client neighbor 10.0.0.80 family inet unicast]
user@R4# set add-path send prefix-policy allow_199
[edit policy-options policy-statement allow_199]
user@R4# set from route-filter 172.16.199.1/32 exact
user@R4# set then accept
```

   - Router R4 can also be configured to send up-to 20 BGP `add-path` routes for a subset of *add-path advertised prefixes*.

```
[edit policy-options policy-statement allow_199]
user@R4# set term match_199 from prefix-list match_199
user@R4# set then add-path send-count 20
```

7. Configure the autonomous system number.

```
[edit routing-options]
user@R4# set autonomous-system 1
```

8. If you are done configuring the device, commit the configuration.

```
user@R4# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R4# show interfaces
fe-1/2/0 {
    unit 41 {
        family inet {
            address 10.0.14.2/24;
        }
    }
}
fe-1/2/1 {
    unit 48 {
        family inet {
            address 10.0.48.1/24;
        }
    }
}
lo0 {
    unit 40 {
        family inet {
            address 10.0.0.40/32;
        }
```

```
    }
}
```

```
user@R4# show protocols
bgp {
    group rr {
        type internal;
        local-address 10.0.0.40;
        family inet {
            unicast {
                add-path {
                    receive;
                }
            }
        }
        neighbor 10.0.0.10;
    }
    group rr_client {
        type internal;
        local-address 10.0.0.40;
        cluster 10.0.0.40;
        neighbor 10.0.0.80 {
            family inet {
                unicast {
                    add-path {
                        send {
                            path-count 6;
                            prefix-policy allow_199;
                        }
                    }
                }
            }
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.40 {
            passive;
        }
        interface fe-1/2/0.41;
```

```
        interface fe-1/2/1.48;
    }
}
```

```
user@R4# show policy-options
policy-statement allow_199 {
    from {
        route-filter 172.16.199.1/32 exact;
    }
    from term match_199 {
        prefix-list match_199;
    }
        then add-path send-count 20;
    then accept;
}
```

```
user@R4# show routing-options
 autonomous-system 1;
```

Configuring Router R5

Step-by-Step Procedure

To configure Router R5:

1. Configure the loopback (lo0) interface and the interface to Router R1.

```
[edit interfaces]
user@R5# set fe-1/2/0 unit 51 family inet address 10.0.15.2/24
user@R5# set lo0 unit 50 family inet address 10.0.0.50/32
```

2. Configure BGP on Router R5's interface.

```
[edit protocols bgp group e1]
user@R5# set type external
user@R5# set neighbor 10.0.15.1 peer-as 1
```

3. Create static routes for redistribution into BGP.

```
[edit routing-options]
user@R5# set static route 172.16.199.1/32 reject
user@R5# set static route 172.16.198.1/32 reject
```

4. Redistribute static and direct routes into BGP.

```
[edit protocols bgp group e1 neighbor 10.0.15.1]
user@R5# set export s2b
[edit policy-options policy-statement s2b]
user@R5# set from protocol static
user@R5# set from protocol direct
user@R5# set then as-path-expand 2
user@R5# set then accept
```

5. Configure the autonomous system number.

```
[edit routing-options]
user@R5# set autonomous-system 2
```

6. If you are done configuring the device, commit the configuration.

```
user@R5# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R5# show interfaces
fe-1/2/0 {
    unit 51 {

        family inet {
            address 10.0.15.2/24;
```

```
        }
    }
}
lo0 {
    unit 50 {
        family inet {
            address 10.0.0.50/32;
        }
    }
}
```

```
user@R5# show protocols
bgp {
    group e1 {
        type external;
        neighbor 10.0.15.1 {
            export s2b;
            peer-as 1;
        }
    }
}
```

```
user@R5# show policy-options
policy-statement s2b {
    from protocol [ static direct ];
    then {
        as-path-expand 2;
        accept;
    }
}
```

```
user@R5# show routing-options
static {
    route 172.16.198.1/32 reject;
    route 172.16.199.1/32 reject;
}
 autonomous-system 2;
```

Configuring Router R6

Step-by-Step Procedure

To configure Router R6:

1. Configure the loopback (lo0) interface and the interface to Router R2.

```
[edit interfaces]
user@R6# set fe-1/2/0 unit 62 family inet address 10.0.26.2/24
user@R6# set lo0 unit 60 family inet address 10.0.0.60/32
```

2. Configure BGP on Router R6's interface.

```
[edit protocols]
user@R6# set bgp group e1 type external
user@R6# set bgp group e1 neighbor 10.0.26.1 peer-as 1
```

3. Create static routes for redistribution into BGP.

```
[edit]
user@R6# set routing-options static route 172.16.199.1/32 reject
user@R6# set routing-options static route 172.16.198.1/32 reject
```

4. Redistribute static and direct routes from Router R6's routing table into BGP.

```
[edit protocols bgp group e1 neighbor 10.0.26.1]
user@R6# set export s2b
[edit policy-options policy-statement s2b]
user@R6# set from protocol static
user@R6# set from protocol direct
user@R6# set then accept
```

5. Configure the autonomous system number.

```
[edit routing-options]
user@R6# set autonomous-system 2
```

6. If you are done configuring the device, commit the configuration.

```
user@R6# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R6# show interfaces
fe-1/2/0 {
    unit 62 {

        family inet {
            address 10.0.26.2/24;
        }
    }
}
lo0 {
    unit 60 {
        family inet {
            address 10.0.0.60/32;
        }
    }
}
```

```
user@R6# show protocols
bgp {
    group e1 {
        type external;
        neighbor 10.0.26.1 {
            export s2b;
            peer-as 1;
        }
```

```
    }
}
```

```
user@R6# show policy-options
policy-statement s2b {
    from protocol [ static direct ];
    then accept;
}
```

```
user@R6# show routing-options
static {
    route 172.16.198.1/32 reject;
    route 172.16.199.1/32 reject;
}
 autonomous-system 2;
```

Configuring Router R7

Step-by-Step Procedure

To configure Router R7:

1. Configure the loopback (lo0) interface and the interface to Router R3.

```
[edit interfaces]
user@R7# set fe-1/2/0 unit 73 family inet address 10.0.37.2/24
user@R7# set lo0 unit 70 family inet address 10.0.0.70/32
```

2. Configure BGP on Router R7's interface.

```
[edit protocols bgp group e1]
user@R7# set type external
user@R7# set neighbor 10.0.37.1 peer-as 1
```

3. Create a static route for redistribution into BGP.

```
[edit]
user@R7# set routing-options static route 172.16.199.1/32 reject
```

4. Redistribute static and direct routes from Router R7's routing table into BGP.

```
[edit protocols bgp group e1 neighbor 10.0.37.1]
user@R7# set export s2b
[edit policy-options policy-statement s2b]
user@R7# set from protocol static
user@R7# set from protocol direct
user@R7# set then accept
```

5. Configure the autonomous system number.

```
[edit routing-options]
user@R7# set autonomous-system 2
```

6. If you are done configuring the device, commit the configuration.

```
user@R7# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R7# show interfaces
fe-1/2/0 {
    unit 73 {

        family inet {
            address 10.0.37.2/24;
        }
    }
```

```
}
lo0 {
    unit 70 {
        family inet {
            address 10.0.0.70/32;
        }
    }
}
```

```
user@R7# show protocols
bgp {
    group e1 {
        type external;
        neighbor 10.0.37.1 {
            export s2b;
            peer-as 1;
        }
    }
}
```

```
user@R7# show policy-options
policy-statement s2b {
    from protocol [ static direct ];
    then accept;
}
```

```
user@R7# show routing-options
static {
    route 172.16.199.1/32 reject;
}
 autonomous-system 2;
```

Configuring Router R8

Step-by-Step Procedure

To configure Router R8:

1. Configure the loopback (lo0) interface and the interface to Router R4.

```
[edit interfaces]
user@R8# set fe-1/2/0 unit 84 family inet address 10.0.48.2/24
user@R8# set lo0 unit 80 family inet address 10.0.0.80/32
```

2. Configure BGP and OSPF on Router R8's interface.

```
[edit protocols]
user@R8# set bgp group rr type internal
user@R8# set bgp group rr local-address 10.0.0.80
user@R8# set ospf area 0.0.0.0 interface lo0.80 passive
user@R8# set ospf area 0.0.0.0 interface fe-1/2/0.84
```

3. Configure Router R8 to receive multiple paths from its neighbor, Router R4.

   The destination of the paths can be any destination that Router R4 can reach through multiple paths.

```
[edit protocols]
user@R8# set bgp group rr neighbor 10.0.0.40 family inet unicast add-path receive
```

4. Configure the autonomous system number.

```
[edit]
user@R8# set routing-options autonomous-system 1
```

5. If you are done configuring the device, commit the configuration.

```
user@R8# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R8# show interfaces
fe-1/2/0 {

    unit 84 {

        family inet {
            address 10.0.48.2/24;
        }
    }
}
lo0 {
    unit 80 {
        family inet {
            address 10.0.0.80/32;
        }
    }
}
```

```
user@R8# show protocols
bgp {
    group rr {
        type internal;
        local-address 10.0.0.80;
        neighbor 10.0.0.40 {
            family inet {
                unicast {
                    add-path {
                        receive;
                    }
                }
            }
        }
    }
}
ospf {
```

```
    area 0.0.0.0 {
        interface lo0.80 {
            passive;
        }
        interface fe-1/2/0.84;
    }
}
```

```
user@R8# show routing-options
 autonomous-system 1;
```

Verification

IN THIS SECTION

- Verifying That the BGP Peers Have the Ability to Send and Receive Multiple Paths | **499**
- Verifying That Router R1 Is Advertising Multiple Paths | **500**
- Verifying That Router R4 Is Receiving and Advertising Multiple Paths | **501**
- Verifying That Router R8 Is Receiving Multiple Paths | **502**
- Checking the Path ID | **503**

Confirm that the configuration is working properly.

Verifying That the BGP Peers Have the Ability to Send and Receive Multiple Paths

Purpose

Make sure that one or both of the following strings appear in the output of the `show bgp neighbor` command:

- `NLRI's for which peer can receive multiple paths: inet-unicast`
- `NLRI's for which peer can send multiple paths: inet-unicast`

Action

```
user@R1> show bgp neighbor 10.0.0.40
Peer: 10.0.0.40+179 AS 1       Local: 10.0.0.10+64227 AS 1
  Type: Internal    State: Established    Flags: <Sync>
...  NLRI's for which peer can receive multiple paths: inet-unicast
...
```

```
user@R4> show bgp neighbor 10.0.0.10
Peer: 10.0.0.10+64227 AS 1     Local: 10.0.0.40+179 AS 1
  Type: Internal    State: Established    Flags: <Sync>
...
  NLRI's for which peer can send multiple paths: inet-unicast
...
```

```
user@R4> show bgp neighbor 10.0.0.80
Peer: 10.0.0.80+55416 AS 1     Local: 10.0.0.40+179 AS 1
  Type: Internal    State: Established  (route reflector client)Flags: <Sync>
  ,,,
  NLRI's for which peer can receive multiple paths: inet-unicast
  ...
```

```
user@R8> show bgp neighbor 10.0.0.40
Peer: 10.0.0.40+179 AS 1       Local: 10.0.0.80+55416 AS 1
  Type: Internal    State: Established    Flags: <Sync>
  ...
  NLRI's for which peer can send multiple paths: inet-unicast
  ...
```

Verifying That Router R1 Is Advertising Multiple Paths

Purpose

Make sure that multiple paths to the 172.16.198.1/32 destination and multiple paths to the 172.16.199.1/32 destination are advertised to Router R4.

Action

```
user@R1> show route advertising-protocol bgp 10.0.0.40
inet.0: 21 destinations, 25 routes (21 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.50/32            10.0.15.2                    100        2 2 I
* 10.0.0.60/32            10.0.0.20                    100        2 I
* 10.0.0.70/32            10.0.0.30                    100        2 I
* 172.16.198.1/32            10.0.0.20                    100        2 I
                          10.0.15.2                    100        2 2 I
* 172.16.199.1/32            10.0.0.20                    100        2 I
                          10.0.0.30                    100        2 I
                          10.0.15.2                    100        2 2 I
* 172.16.200.0/30            10.0.0.20                    100        2 I
```

Meaning

When you see one prefix and more than one next hop, it means that multiple paths are advertised to Router R4.

Verifying That Router R4 Is Receiving and Advertising Multiple Paths

Purpose

Make sure that multiple paths to the 172.16.199.1/32 destination are received from Router R1 and advertised to Router R8. Make sure that multiple paths to the 172.16.198.1/32 destination are received from Router R1, but only one path to this destination is advertised to Router R8.

Action

```
user@R4> show route receive-protocol bgp 10.0.0.10
inet.0: 19 destinations, 22 routes (19 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.50/32            10.0.15.2                    100        2 2 I
* 10.0.0.60/32            10.0.0.20                    100        2 I
* 10.0.0.70/32            10.0.0.30                    100        2 I
* 172.16.198.1/32            10.0.0.20                    100        2 I
                          10.0.15.2                    100        2 2 I
* 172.16.199.1/32            10.0.0.20                    100        2 I
```

```
                        10.0.0.30                100        2 I
                        10.0.15.2                100        2 2 I
* 172.16.200.0/30          10.0.0.20                100        2 I
```

```
user@R4> show route advertising-protocol bgp 10.0.0.80
inet.0: 19 destinations, 22 routes (19 active, 0 holddown, 0 hidden)
  Prefix                 Nexthop              MED     Lclpref    AS path
* 10.0.0.50/32           10.0.15.2                    100        2 2 I
* 10.0.0.60/32           10.0.0.20                    100        2 I
* 10.0.0.70/32           10.0.0.30                    100        2 I
* 172.16.198.1/32           10.0.0.20                    100        2 I
* 172.16.199.1/32           10.0.0.20                    100        2 I
                         10.0.0.30                    100        2 I
                         10.0.15.2                    100        2 2 I
* 172.16.200.0/30           10.0.0.20                    100        2 I
```

Meaning

The `show route receive-protocol` command shows that Router R4 receives two paths to the 172.16.198.1/32 destination and three paths to the 172.16.199.1/32 destination. The `show route advertising-protocol` command shows that Router R4 advertises only one path to the 172.16.198.1/32 destination and advertises all three paths to the 172.16.199.1/32 destination.

Because of the prefix policy that is applied to Router R4, Router R4 does not advertise multiple paths to the 172.16.198.1/32 destination. Router R4 advertises only one path to the 172.16.198.1/32 destination even though it receives multiple paths to this destination.

Verifying That Router R8 Is Receiving Multiple Paths

Purpose

Make sure that Router R8 receives multiple paths to the 172.16.199.1/32 destination through Router R4. Make sure that Router R8 receives only one path to the 172.16.198.1/32 destination through Router R4.

Action

```
user@R8> show route receive-protocol bgp 10.0.0.40
inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.50/32            10.0.15.2                    100        2 2 I
* 10.0.0.60/32            10.0.0.20                    100        2 I
* 10.0.0.70/32            10.0.0.30                    100        2 I
* 172.16.198.1/32            10.0.0.20                    100        2 I
* 172.16.199.1/32            10.0.0.20                    100        2 I
                          10.0.0.30                    100        2 I
                          10.0.15.2                    100        2 2 I
* 200.1.1.0/30            10.0.0.20                    100        2 I
```

Checking the Path ID

Purpose

On the downstream devices, Router R4 and Router R8, verify that a path ID uniquely identifies the path. Look for the `Addpath Path ID:` string.

Action

```
user@R4> show route 172.16.199.1/32 detail

inet.0: 18 destinations, 20 routes (18 active, 0 holddown, 0 hidden)
172.16.199.1/32 (3 entries, 3 announced)
        *BGP    Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 9
                Source: 10.0.0.10
                Next hop type: Router, Next hop index: 676
                Next hop: 10.0.14.1 via lt-1/2/0.41, selected
                Protocol next hop: 10.0.0.20
                Indirect next hop: 92041c8 262146
                State: <Active Int Ext>
                Local AS:     1 Peer AS:     1
                Age: 1:44:37    Metric2: 2
                Task: BGP_1.10.0.0.10+64227
                Announcement bits (3): 2-KRT 3-BGP RT Background 4-Resolve tree 1
```

```
                AS path: 2 I (Originator) Cluster list:  10.0.0.10
                AS path:  Originator ID: 10.0.0.20
                Accepted
                Localpref: 100
                Router ID: 10.0.0.10
                Addpath Path ID: 1
        BGP     Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 10.0.0.10
                Next hop type: Router, Next hop index: 676
                Next hop: 10.0.14.1 via lt-1/2/0.41, selected
                Protocol next hop: 10.0.0.30
                Indirect next hop: 92042ac 262151
                State: <NotBest Int Ext>
                Inactive reason: Not Best in its group - Router ID
                Local AS:     1 Peer AS:     1
                Age: 1:44:37    Metric2: 2
                Task: BGP_1.10.0.0.10+64227
                Announcement bits (1): 3-BGP RT Background
                AS path: 2 I (Originator) Cluster list:  10.0.0.10
                AS path:  Originator ID: 10.0.0.30
                Accepted
                Localpref: 100
                Router ID: 10.0.0.10
                Addpath Path ID: 2
        BGP     Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 10.0.0.10
                Next hop type: Router, Next hop index: 676
                Next hop: 10.0.14.1 via lt-1/2/0.41, selected
                Protocol next hop: 10.0.15.2
                Indirect next hop: 92040e4 262150
                State: <Int Ext>
                Inactive reason: AS path
                Local AS:     1 Peer AS:     1
                Age: 1:44:37    Metric2: 2
                Task: BGP_1.10.0.0.10+64227
                Announcement bits (1): 3-BGP RT Background
                AS path: 2 2 I
                Accepted
                Localpref: 100
```

```
                Router ID: 10.0.0.10
                Addpath Path ID: 3
```

```
user@R8> show route 172.16.199.1/32 detail

inet.0: 17 destinations, 19 routes (17 active, 0 holddown, 0 hidden)
172.16.199.1/32 (3 entries, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 9
                Source: 10.0.0.40
                Next hop type: Router, Next hop index: 1045
                Next hop: 10.0.48.1 via lt-1/2/0.84, selected
                Protocol next hop: 10.0.0.20
                Indirect next hop: 91fc0e4 262148
                State: <Active Int Ext>
                Local AS:     1 Peer AS:     1
                Age: 1:56:51    Metric2: 3
                Task: BGP_1.10.0.0.40+179
                Announcement bits (2): 2-KRT 4-Resolve tree 1
                AS path: 2 I (Originator) Cluster list:  10.0.0.40 10.0.0.10
                AS path:  Originator ID: 10.0.0.20
                Accepted
                Localpref: 100
                Router ID: 10.0.0.40
                Addpath Path ID: 1
         BGP    Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 10.0.0.40
                Next hop type: Router, Next hop index: 1045
                Next hop: 10.0.48.1 via lt-1/2/0.84, selected
                Protocol next hop: 10.0.0.30
                Indirect next hop: 91fc1c8 262152
                State: <NotBest Int Ext>
                Inactive reason: Not Best in its group - Router ID
                Local AS:     1 Peer AS:     1
                Age: 1:56:51    Metric2: 3
                Task: BGP_1.10.0.0.40+179
                AS path: 2 I (Originator) Cluster list:  10.0.0.40 10.0.0.10
                AS path:  Originator ID: 10.0.0.30
```

```
                Accepted
                Localpref: 100
                Router ID: 10.0.0.40
                Addpath Path ID: 2
          BGP   Preference: 170/-101
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 10.0.0.40
                Next hop type: Router, Next hop index: 1045
                Next hop: 10.0.48.1 via lt-1/2/0.84, selected
                Protocol next hop: 10.0.15.2
                Indirect next hop: 91fc2ac 262153
                State: <Int Ext>
                Inactive reason: AS path
                Local AS:     1 Peer AS:     1
                Age: 1:56:51    Metric2: 3
                Task: BGP_1.10.0.0.40+179
                AS path: 2 2 I (Originator) Cluster list:  10.0.0.40
                AS path:  Originator ID: 10.0.0.10
                Accepted
                Localpref: 100
                Router ID: 10.0.0.40
                Addpath Path ID: 3
```

RELATED DOCUMENTATION

- Understanding the Advertisement of Multiple Paths to a Single Destination in BGP
- Understanding Adding AS Numbers to BGP AS Paths | **468**

Improve the Performance of AS Path Lookup in BGP Policy

SUMMARY

IN THIS SECTION

- AS Path Lookup in a BGP Policy Without Regular Expression Overview | **507**

- Configure AS Path Lookup Without Using Regular Expression | 508

AS Path Lookup in a BGP Policy Without Regular Expression Overview

IN THIS SECTION

- Benefits of AS-Path without using regular expression in BGP policy: | 507

When working with BGP AS paths and routing policy match conditions, you can configure BGP policies to check for an autonomous system (AS) match in an AS path without using regular expressions. The BGP policy compares the AS to an AS-list or As-list-group and returns true if it finds a match. You can configure the BGP policy to check for a matching origin, neighbor, or transit AS. This feature provides a faster alternative to match origin, transit, and peer AS numbers than using a regular expression.

Benefits of AS-Path without using regular expression in BGP policy:

- Optimized lookup for origin, neighbor, transit ASes improves the performance.
- Provides a faster lookup in terms of speed.

The following operations to match the ASes in an AS path are supported:

- **Match the Originating AS in the AS Path**—Compares the AS that originated the route. Evaluates if the right most AS number on the AS path belongs to the `as-list` or `as-list-group` specified in the `as-path-origins` configuration statement at the `[edit policy-options policy-statement` *`policy-name`* `from]` hierarchy level. In the case where the route has been aggregated, and the location of the originating AS contains an AS-set, the `as-path-origins` operator evaluates to true if any AS contained in the AS-set belongs to the `as-list` or `as-list-group` specified in the `as-path-origins` configuration statement.
- **Match the Neighbor AS in the AS Path**—Compares the neighbor AS in the AS path. Evaluates if the first AS number on the AS path matches the `as-list` or `as-list-group` specified in the `as-path-neighbors` configuration statement at the `[edit policy-options policy-statement` *`policy-name`* `from]` hierarchy level. If the neighboring AS location happens to be an AS-set, the `as-path-neighbors` operator evaluates to true if any AS contained in the AS-set belongs to the `as-list` or `as-list-group` specified in the `as-path-neighbors` configuration statement.

- **Match the Transit AS in the AS Path**—Compares any AS in the AS-Path. Evaluates when any AS belongs to the `as-list` or `as-list-group` specified in the `as-path-transit` configuration statement at the `[edit policy-options policy-statement` *`policy-name`* `from]` hierarchy level. In the case of AS-set, the as-path-transit operator compares all the ASes in the AS-set.

Configure AS Path Lookup Without Using Regular Expression

You can configure AS path lookups by defining AS lists, or AS list groups for origin, neighbor, and transit ASes and filter the routes without using regular expression.

The table here shows configurations of universal match based on regular expressions and equivalent match with faster execution time.

Match type	Universal match based on regular expressions	Equivalent match with faster execution time
Peer	set policy-options as-path peer-match "^101.*"	set policy-options as-list peer-match members 101
Transit	set policy-options as-path transit-match ".*61453.*10001.*40007$"	set policy-options as-list transit-match members 61453
Origin	set policy-options as-path origin-match ".*54367$"	set policy-options as-list origin-match members 54367

The following sample configuration shows how you can define AS lists (`as-list` *`as-list-name`*) for origin, neighbor, and transit ASes and how you can use policies to filter the routes without using regular expression:

Step 1: Define AS lists for matching origin, neighbor, and transit AS and apply it as a filter to filter the routes using policies.

```
set policy-options as-list origin-match members 54367
set policy-options as-list neighbor-match members 101
set policy-options as-list transit-match members 61453
set policy-options as-list transit-match members 10001
```

NOTE: You can also define AS list groups (`as-list-group` *`group-name`*) for matching origin, neighbor, and transit ASes to filter the routes using policies. The following is a sample configuration to define AS list groups to match origin ASes to filter the routes using policies:

```
set policy-options as-list-group origin_group as-list origin-match-1 members 3-4
set policy-options as-list-group origin_group as-list origin-match-2 members 6-9
set policy-options policy-statement neighbor-accept term 1 from as-path-origins as-list-group
```

```
origin_group
set policy-options policy-statement neighbor-accept term 1 then accept
set policy-options policy-statement neighbor-accept term 2 then reject
```

NOTE: AS list groups to match the origin, neighbor, and transit ASes could be an AS member (for example, 101) or a range of AS members (for example, 6-9). In this case, all the routes originating from 6, 7, 8, 9 will be matched.

If you are using a range of AS members (as-start to as-finish), then the as-start member value should be less than or equal to as-finish member value. The AS member or the AS member range (as-start to as-finish) cannot be 0.

The `as-list` or `as-list-group` defines an AS set.

While performing an AS set lookup for origins and neighbors, the first or last AS from an AS path is matched. In the case of transits, there could be multiple iterations on the AS path to perform AS set lookup.

Step 2: Configure policies to match and filter routes based on origin, neighbor, and transit ASes.

```
set policy-options policy-statement as-list-match term transit-match from as-path-transits as-
list transit-match
set policy-options policy-statement as-list-match term transit-match then local-preference 300
set policy-options policy-statement as-list-match term transit-match then accept
set policy-options policy-statement as-list-match term origin-match from as-path-origins as-list
origin-match
set policy-options policy-statement as-list-match term origin-match then local-preference 400
set policy-options policy-statement as-list-match term origin-match then accept
set policy-options policy-statement as-list-match term neighbor-match from as-path-neighbors as-
list peer-match
set policy-options policy-statement as-list-match term peer-match then local-preference 200
set policy-options policy-statement as-list-match term peer-match then accept
```

Step 3: Define the local autonomous system.

```
set routing-options autonomous-system 100
```

Step 4: Apply the policy as BGP import policy to filter the routes.

```
set protocols bgp group ebgp-1 type external
set protocols bgp group ebgp-1 import as-list-match
set protocols bgp group ebgp-1 family inet unicast
set protocols bgp group ebgp-1 neighbor 192.168.1.2 peer-as 101
set protocols bgp group ebgp-1 neighbor 192.168.1.6 peer-as 102
```

NOTE: This policy can be applied as an import or export policy to filter the routes to take corresponding action defined in the policy.

You can use the `show route` CLI command to view the routes in the routing table.

NOTE: The configuration statements in the from clause match condition occurs in both `[edit policy-options policy-statement` *`policy-name`* `from]` and `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `from]` hierarchy levels.

CHAPTER 6

Configuring Communities as Match Conditions

IN THIS CHAPTER

- Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions | **511**
- Understanding How to Define BGP Communities and Extended Communities | **513**
- How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions | **521**
- Example: Configuring Communities in a Routing Policy | **528**
- Example: Configuring Extended Communities in a Routing Policy | **548**
- Example: Configuring BGP Large Communities | **560**
- Example: Configuring a Routing Policy Based on the Number of BGP Communities | **572**
- Example: Configuring a Routing Policy That Removes BGP Communities | **583**

Understanding BGP Communities, Extended Communities, and Large 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 enables you to perform actions on a group without having to elaborate upon each member. You can use community and extended communities attributes to trigger routing decisions, such as acceptance, rejection, preference, or redistribution.

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.

A community value is a 32-bit field that is divided into two main sections. The first 16 bits of the value encode the AS number of the network that originated the community, while the last 16 bits carry a unique number assigned by the AS. This system attempts to guarantee a globally unique set of community values for each AS in the Internet. Junos OS uses a notation of *as-number*:*community-value*, where each value is a decimal number. The AS values of 0 and 65,535 are reserved, as are all of the

community values within those AS numbers. Each community, or set of communities, is given a name within the [edit policy-options] configuration hierarchy. The name of the community uniquely identifies it to the routing device and serves as the method by which routes are categorized. For example, a route with a community value of 64510:1111 might belong to the community named AS64510-routes. The community name is also used within a routing policy as a match criterion or as an action. The command syntax for creating a community is: policy-options community *name* members [*community-ids*]. The *community-ids* are either a single community value or multiple community values. When more than one value is assigned to a community name, the routing device interprets this as a logical AND of the community values. In other words, a route must have all of the configured values before being assigned the community name.

The regular community attribute is four octets. Networking enhancements, such as VPNs, have functionality requirements that can be satisfied by an attribute such as a community. However, the 4-octet community value does not provide enough expansion and flexibility to accommodate VPN requirements. This leads to the creation of extended communities. An extended community is an 8-octet value that is also divided into two main sections. The first 2 octets of the community encode a type field while the last 6 octets carry a unique set of data in a format defined by the type field. Extended communities provide a larger range for grouping or categorizing communities.

The BGP extended communities attribute format has three fields: *type*:*administrator*:*assigned-number*. The routing device expects you to use the words target or origin to represent the type field. The administrator field uses a decimal number for the AS or an IPv4 address, while the assigned number field expects a decimal number no larger than the size of the field (65,535 for 2 octets or 4,294,967,295 for 4 octets).

When specifying community IDs for standard and extended community attributes, you can use UNIX-style regular expressions. The only exception is for VPN import policies (vrf-import), which do not support regular expressions for the extended communities attribute.

Regular BGP communities attributes are a variable length attribute consisting of a set of one or more 4-byte values that was split into 16 bit values. The most significant word is interpreted as an AS number and least significant word is a locally defined value assigned by the operator of the AS. Since the adoption of 4-byte ASNs, the 4-byte BGP regular community and 6-byte BGP extended community can no longer support BGP community attributes. Operators often encode AS number in the local portion of the BGP community that means that sometimes the format of the community is ASN:ASN. With the 4-byte ASN , you need 8-bytes to encode it. Although BGP extended community permits a 4-byte AS to be encoded as the global administrator field, the local administrator field has only 2-byte of available space. Thus, 6-byte extended community attribute is also unsuitable. To overcome this, Junos OS allows you to configure optional transitive path attribute - a 12-byte BGP large community that provides the most significant 4-byte value to encode autonomous system number as the global administrator and the remaining two 4-byte assigned numbers to encode the local values as defined in RFC 8092. You can configure BGP large community at the [edit policy-options community *community-name* members] and [edit routing-options static route *ip-address* community] hierarchy levels. The BGP large community attributes format has four fields: large:*global administrator*:*assigned number*:*assigned number*.

The BGP IPv6 unicast address specific extended community are encoded as a set of 20-bytes value. The 20-byte value gets interpreted in the following format:

- Most significant 2-bytes encodes the Type and Sub-Type value (high value (most significant byte) and Low value (second most significant byte)).
- Next 16-bytes encodes the IPv6 unicast address. It is the global administrator in the IETF RFC.
- Last 2-bytes encodes the operator defined local values. It is local administrator in the IETF RFC.

The IPv6 unicast address specific BGP extended community attributes are represented by a keyword `ipv6-target`, `ipv6-origin`, or `ipv6-extended` followed by IPv6 and local administrator separated by <, >, and :.

NOTE: The length of the BGP large communities attribute value should be a non-zero multiple of 12.

RELATED DOCUMENTATION

Understanding How to Define BGP Communities and Extended Communities | 513

How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions | 521

Example: Configuring a Routing Policy That Removes BGP Communities

Example: Configuring Communities in a Routing Policy | 528

Example: Configuring Extended Communities in a Routing Policy | 548

Understanding How to Define BGP Communities and Extended Communities

IN THIS SECTION

- Defining BGP Communities for Use in Routing Policy Match Conditions | 514
- Defining BGP Extended Communities for Use in Routing Policy Match Conditions | 519

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

Defining BGP Communities for Use in Routing Policy Match Conditions

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

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

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

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

```
as-number:community-value;
```

- *as-number*—AS number of the community member. It can be a value from 0 through 65,535. You can use the following notation in specifying the AS number:
  - String of digits.
  - Asterisk (*)—A wildcard character that matches all AS numbers. (In the definition of the community attribute, the asterisk also functions as described in Table 21 on page 516.)
  - 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 516.
- *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 21 on page 516.)

- 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. This will tag the routes you specify in `[policy-options policy-statement]` with the configured name or community value. In a separate configuration, you also need to create a filter for the imported routes in your BGP import policy.

- 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. A stand alone autonomous system that is not part of a confederation should be considered a confederation itself.
- 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.

You can include the following IPv6 unicast address community names (defined in RFC 5701, BGP Communities Attribute) to accommodate IPv6 unicast address specific extended community:

```
[edit policy-options]
policy-statement send-direct {
term 1 {
        then {
            community add community-name;
            community add community-name;
            community add community-name;
            accept;
        }
    }
}
community community-name members [ ipv6-target:<IPv6 unicast address>:operator-defined local values ipv6-
target:<IPv6 unicast address>:operator-defined local values ];
community community-name members [ ipv6-origin:<IPv6 unicast address>:operator-defined local values ipv6-
origin:<IPv6 unicast address>:operator-defined local values ];
community community-name members [ ipv6-extended:type-and-subtype value:<IPv6 unicast address>:operator-
defined local values ipv6-extended:type-and-subtype value:<IPv6 unicast address>:operator-defined
local values ];
```

ipv6-target identifies the VPN IPv6 target unicast address used in a policy match. *ipv6-origin* identifies the source of the IPv6 unicast address in a policy match. *ipv6-extended* identifies the extended format of the IPv6 unicast address in a policy match.

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 21 on page 516 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 22 on page 518 shows examples of how to define *community-ids* using community regular expressions. The operator is optional.

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

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

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

Table 21: Community Attribute Regular Expression Operators

Operator	Match Definition
{*m*,*n*}	At least *m* and at most *n* repetitions of *term*. Both *m* and *n* must be positive integers, and *m* must be smaller than *n*.

Table 21: Community Attribute Regular Expression Operators *(Continued)*

Operator	Match Definition
{*m*}	Exactly *m* repetitions of *term*. *m* must be a positive integer.
{*m*,}	*m* or more repetitions of *term*. *m* must be a positive integer.
*	Zero or more repetitions of *term*. This is equivalent to {0,}.
+	One or more repetitions of *term*. This is equivalent to {1,}.
?	Zero or one repetition of *term*. This is equivalent to {0,1}.
|	One of the two terms on either side of the pipe.
-	Between a starting and ending range, inclusive.
^	Character at the beginning of a community attribute regular expression.
$	Character at the end of a community attribute regular expression.
[ ]	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 ([).

Table 21: Community Attribute Regular Expression Operators *(Continued)*

Operator	Match Definition
( )	Group of terms that are enclosed in parentheses. If enclosed in quotation marks with no intervening space ("()"), indicates a null. Intervening space between the parentheses and the terms is ignored.
“ ”	Characters (such as space, tab, question mark, and bracket) that are enclosed within quotation marks in a community attribute regular expression indicate special characters.

Table 22: 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 78:64500
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 34:64509
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:

```
[edit policy-options]
community name {
    members [ community-ids ];
}
```

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

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

```
type:administrator:assigned-number
```

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

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

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

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

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

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

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

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

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

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

`assigned-number` identifies the local provider.

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

> **NOTE**: 4-byte ASes can be specified only as a part of extended communities and hence the letter 'L' is not allowed in a base BGP regular expression community. For example, to allow matches against an extended community, use extended community expressions like `origin:334324L:*` and `target:334324L:*` instead of `334324L:*`

In Junos OS Release 9.2 and later, you can also use AS-dot notation when defining a 4-byte AS number for the `target` and `origin` extended communities. Specify two integers joined by a period: *16-bit high-order value in decimal.16-bit low-order value in decimal*. For example, the 4-byte AS number represented in plain-number format as 65546 is represented in AS-dot notation as 1.10.

Examples: Defining BGP Extended Communities

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

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

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

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

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

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

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

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

RELATED DOCUMENTATION

Example: Configuring Communities in a Routing Policy | 528

Example: Configuring Extended Communities in a Routing Policy | 548

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

IN THIS SECTION

- Multiple Matches | 523
- Inverting Community Matches | 525
- Extended Community Type | 525
- Multiple Communities Are Matched with Ex-OR Logic | 526
- Including BGP Communities and Extended Communities in Routing Policy Match Conditions | 527

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. Each member is evaluated against each community associated with the route. (Communities are an unordered property of a route. For example, 1:2 3:4 is the same as 3:4 1:2.) Only one community from the route is required to match for the member evaluation to be successful.
- Community regular expressions are evaluated on a character-by-character basis. For example, if a route contains community 1234:5678, the regular expressions see nine discrete characters, including the colon (:), instead of two sets of numbers (1234 and 5678) separated by a colon. For example:

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

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

For example:

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

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

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

- To match `comm-two`, the route must have a community that matches 7:8 and a community that matches 9:10.

Multiple Matches

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

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

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

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

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

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

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

Both methods are shown in the following examples.

Be More Specific in the Regular Expression

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

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

Specify the Site of Origin in the Community Name

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

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

Inverting Community Matches

The `community` match condition defines a regular expression and if it matches the community attribute of the received prefix, Junos OS returns a TRUE result. If not, Junos OS returns a FALSE result. The `invert-match` statement makes Junos OS behave to the contrary. If there is a match, Junos OS returns a FALSE result. If there is no match, Junos OS returns a TRUE result. To invert the results of the community expression matching, include the `invert-match` statement in the community configuration.

```
[edit policy-options community name]
invert-match;
```

Extended Community Type

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

Communities:

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

Community attribute:

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

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

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

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

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

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

NOTE: With the implementation of the large community feature in release 17.3, Junos OS checks to prevent the matching of extended or large communities against base BGP or base BGP regular expression communities. In other words, a received community can only be matched against the appropriate wild community pattern, like normal communities against simple-wild, extended communities against extended-wild and large communities against large-wild patterns. For example, to allow the advertisement of routes carrying the origin extended community, use the `origin:*:65020` expression.

Multiple Communities Are Matched with Ex-OR Logic

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

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

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

Both labels are aggregated, as shown here:

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

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

A more complete example of community values is shown here:

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

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

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 ];
}
```

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.

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).

RELATED DOCUMENTATION

- Using UNIX Regular Expressions in Community Names | 516
- Example: Configuring Communities in a Routing Policy | 528
- Example: Configuring Extended Communities in a Routing Policy | 548
- Example: Configuring a Routing Policy That Removes BGP Communities | 583
- Example: Configuring a Routing Policy Based on the Number of BGP Communities | 572

Example: Configuring Communities in a Routing Policy

IN THIS SECTION

- Requirements | 528
- Overview | 529
- Configuration | 531
- Verification | 543

A community is a route attribute used by BGP to administratively group routes with similar properties.

Requirements

No special configuration beyond device initialization is required before configuring this example.

- Updated and revalidated using vMX on Junos OS Release 21.1R1.

Overview

IN THIS SECTION

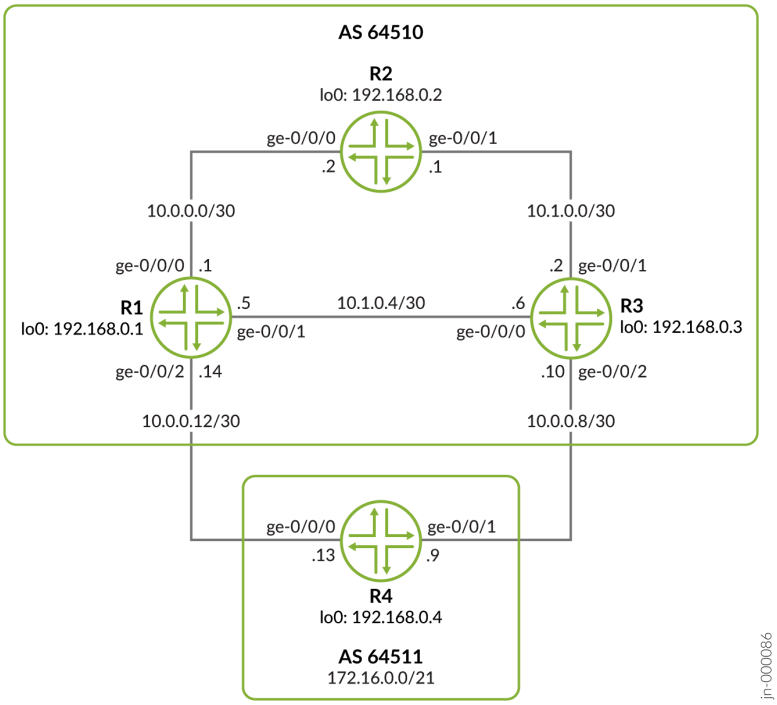
- Topology | **530**

One main role of the community attribute is to be an administrative tag value used to associate routes together. Generally, these routes share some common properties, but that is not required. Communities are a flexible tool within BGP. An individual community value can be assigned to a single route or multiple routes. A route can be assigned a single community value or multiple values. Networks use the community attribute to assist in implementing administrative routing policies. A route's assigned value can allow it to be accepted into the network, or rejected from the network, or allow it to modify attributes.

Figure 32 on page 530 shows device R1, device R2, and device R3 as internal BGP (IBGP) peers in autonomous system (AS) 64510. Device R4 is advertising the 172.16.0.0/21 address space from AS 64511.

Topology

Figure 32: Topology for Regular BGP Communities



The specific routes received by device R1 from device R4 are as follows:

```
user@R1> show route receive-protocol bgp 10.0.0.13

inet.0: 24 destinations, 28 routes (24 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.0.0/24           10.0.0.13                               64511 I
* 172.16.1.0/24           10.0.0.13                               64511 I
* 172.16.2.0/24           10.0.0.13                               64511 I
* 172.16.3.0/24           10.0.0.13                               64511 I
  172.16.4.0/24           10.0.0.13                               64511 I
  172.16.5.0/24           10.0.0.13                               64511 I
  172.16.6.0/24           10.0.0.13                               64511 I
  172.16.7.0/24           10.0.0.13                               64511 I
```

The administrators of AS 64511 want to receive certain user traffic from device R1, and other user traffic from device R3. To accomplish this administrative goal, device R4 attaches the community value

of 64511:1 to some routes that it sends and attaches the community value 64511:3 to other routes that it sends. Routing policies within AS 64510 are configured using a community match criterion to change the local preference of the received routes to new values that alter the BGP route selection algorithm. The route with the highest local preference value is preferred.

On device R1, routes with the 64511:1 community value are assigned a local preference of 200, and routes with the 64511:3 community value are assigned a local preference of 50. On device R3, the reverse is done so that routes with the 64511:3 community value are assigned a local preference of 200, and routes with the 64511:1 community value are assigned a local preference of 50. This information is then communicated through IBGP by both device R1 and device R3 to device R2.

"CLI Quick Configuration" on page 531 shows the configuration for all of the devices in Figure 32 on page 530.

The section "Step By Step Configuration" on page 534 describes the configuration steps on devices R1 and R4.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **531**

CLI Quick Configuration

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

Device R1

```
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces ge-0/0/1 unit 0 family inet address 10.1.0.5/30
set interfaces ge-0/0/2 unit 0 family inet address 10.0.0.14/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set policy-options policy-statement change-local-preference term find-R1-routes from community
R1_PREFERRED
set policy-options policy-statement change-local-preference term find-R1-routes then local-
preference 200
set policy-options policy-statement change-local-preference term find-R3-routes from community
```

```
R3_PREFERRED
set policy-options policy-statement change-local-preference term find-R3-routes then local-
preference 50
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.0.0.12/30 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options community R3_PREFERRED members 64511:3
set policy-options community R1_PREFERRED members 64511:1
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.1
set protocols bgp group int export send-direct
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group int neighbor 192.168.0.3
set protocols bgp group ext type external
set protocols bgp group ext import change-local-preference
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.13
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device R2

```
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.2/30
set interfaces ge-0/0/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.2
set protocols bgp group int neighbor 192.168.0.1
set protocols bgp group int neighbor 192.168.0.3
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64510
```

Device R3

```
set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.6/30
set interfaces ge-0/0/1 unit 0 family inet address 10.1.0.2/30
set interfaces ge-0/0/2 unit 0 family inet address 10.0.0.10/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set policy-options policy-statement change-local-preference term find-R3-routes from community
R3_PREFERRED
set policy-options policy-statement change-local-preference term find-R3-routes then local-
preference 200
set policy-options policy-statement change-local-preference term find-R1-routes from community
R1_PREFERRED
set policy-options policy-statement change-local-preference term find-R1-routes then local-
preference 50
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 from route-filter 10.0.0.8/30 exact
set policy-options policy-statement send-direct term 1 then accept
set policy-options community R1_PREFERRED members 64511:1
set policy-options community R3_PREFERRED members 64511:3
set protocols bgp group int type internal
set protocols bgp group int local-address 192.168.0.3
set protocols bgp group int export send-direct
set protocols bgp group int neighbor 192.168.0.1
set protocols bgp group int neighbor 192.168.0.2
set protocols bgp group ext type external
set protocols bgp group ext import change-local-preference
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.9
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64510
```

Device R4

```
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.13/30
set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.9/30
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
```

```
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 from route-filter 172.16.0.0/24 exact
set policy-options policy-statement send-static term 1 from route-filter 172.16.1.0/24 exact
set policy-options policy-statement send-static term 1 from route-filter 172.16.2.0/24 exact
set policy-options policy-statement send-static term 1 from route-filter 172.16.3.0/24 exact
set policy-options policy-statement send-static term 1 then community add R1_PREFERRED
set policy-options policy-statement send-static term 1 then accept
set policy-options policy-statement send-static term 2 from protocol static
set policy-options policy-statement send-static term 2 from route-filter 172.16.4.0/24 exact
set policy-options policy-statement send-static term 2 from route-filter 172.16.5.0/24 exact
set policy-options policy-statement send-static term 2 from route-filter 172.16.6.0/24 exact
set policy-options policy-statement send-static term 2 from route-filter 172.16.7.0/24 exact
set policy-options policy-statement send-static term 2 then community add R3_PREFERRED
set policy-options policy-statement send-static term 2 then accept
set policy-options policy-statement send-static term 3 then reject
set policy-options community R3_PREFERRED members 64511:3
set policy-options community R1_PREFERRED members 64511:1
set protocols bgp group to-R1 type external
set protocols bgp group to-R1 export send-static
set protocols bgp group to-R1 peer-as 64510
set protocols bgp group to-R1 neighbor 10.0.0.14
set protocols bgp group to-R3 type external
set protocols bgp group to-R3 export send-static
set protocols bgp group to-R3 peer-as 64510
set protocols bgp group to-R3 neighbor 10.0.0.10
set routing-options router-id 192.168.0.4
set routing-options autonomous-system 64511
set routing-options static route 172.16.0.0/24 reject
set routing-options static route 172.16.1.0/24 reject
set routing-options static route 172.16.2.0/24 reject
set routing-options static route 172.16.3.0/24 reject
set routing-options static route 172.16.4.0/24 reject
set routing-options static route 172.16.5.0/24 reject
set routing-options static route 172.16.6.0/24 reject
set routing-options static route 172.16.7.0/24 reject
```

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure device R1:

1. Configure the interfaces.

```
[edit interfaces]
user@R1# set ge-0/0/0 unit 0 family inet address 10.0.0.1/30
user@R1# set ge-0/0/1 unit 0 family inet address 10.1.0.5/30
user@R1# set ge-0/0/2 unit 0 family inet address 10.0.0.14/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure internal gateway protocol (IGP) connections to devices R2 and R3.

```
[edit protocols ospf area 0.0.0.0]
user@R1# set interface ge-0/0/0.0
user@R1# set interface ge-0/0/1.0
user@R1# set interface lo0.0 passive
```

3. Configure the IBGP connections to devices R2 and R3.

```
[edit protocols bgp group int]
user@R1# set type internal
user@R1# set local-address 192.168.0.1
user@R1# set export send-direct
user@R1# set neighbor 192.168.0.2
user@R1# set neighbor 192.168.0.3
```

4. Configure the EBGP connection to device R4.

```
[edit protocols bgp group ext]
user@R1# set type external
user@R1# set import change-local-preference
user@R1# set peer-as 64511
user@R1# set neighbor 10.0.0.13
```

5. Configure the policy `send-direct`.

This policy is referenced in the IBGP configuration and enables device R2 to have external reachability. An alternative is to configure a `next-hop self` policy on device R1 and device R3.

```
[edit policy-options policy-statement send-direct term 1]
user@R1# set from protocol direct
user@R1# set from route-filter 10.0.0.12/30 exact
user@R1# set then accept
```

6. Configure the policy that changes the local preference for routes with specified community tags.

```
[edit policy-options ]
user@R1# set policy-statement change-local-preference term find-R1-routes from community
R1_PREFERRED
user@R1# set policy-statement change-local-preference term find-R1-routes then local-
preference 200
user@R1# set policy-statement change-local-preference term find-R3-routes from community
R3_PREFERRED
user@R1# set policy-statement change-local-preference term find-R3-routes then local-
preference 50
user@R1# set community R3_PREFERRED members 64511:3
user@R1# set community R1_PREFERRED members 64511:1
```

7. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R1# set router-id 192.168.0.1
user@R1# set autonomous-system 64510
```

To configure device R4:

1. Configure the interfaces.

```
[edit interfaces]
user@R4# set ge-0/0/0 unit 0 family inet address 10.0.0.13/30
user@R4# set ge-0/0/1 unit 0 family inet address 10.0.0.9/30
user@R4# set lo0 unit 0 family inet address 192.168.0.4/32
```

2. Configure the EBGP connection to device R1 and device R3.

```
[edit protocols bgp]
user@R4# set group to-R1 type external
user@R4# set group to-R1 export send-static
user@R4# set group to-R1 peer-as 64510
user@R4# set group to-R1 neighbor 10.0.0.14
user@R4# set group to-R3 type external
user@R4# set group to-R3 export send-static
user@R4# set group to-R3 peer-as 64510
user@R4# set group to-R3 neighbor 10.0.0.10
```

3. Configure the community tags.

```
[edit policy-options ]
user@R4# set community R3_PREFERRED members 64511:3
user@R4# set community R1_PREFERRED members 64511:1
```

4. Configure the policy `send-static`.

   This policy is referenced in the EBGP connections to device R1 and device R3. The policy attaches the 64511:1 (PREFERRED) community to some routes and the 64511:3 (NOT_PREFERRED) community to other routes.

```
[edit policy-options]
user@R4# set policy-statement send-static term 1 from protocol static
user@R4# set policy-statement send-static term 1 from route-filter 172.16.0.0/24 exact
user@R4# set policy-statement send-static term 1 from route-filter 172.16.1.0/24 exact
user@R4# set policy-statement send-static term 1 from route-filter 172.16.2.0/24 exact
user@R4# set policy-statement send-static term 1 from route-filter 172.16.3.0/24 exact
user@R4# set policy-statement send-static term 1 then community add R1_PREFERRED
user@R4# set policy-statement send-static term 1 then accept
user@R4# set policy-statement send-static term 2 from protocol static
user@R4# set policy-statement send-static term 2 from route-filter 172.16.4.0/24 exact
user@R4# set policy-statement send-static term 2 from route-filter 172.16.5.0/24 exact
user@R4# set policy-statement send-static term 2 from route-filter 172.16.6.0/24 exact
user@R4# set policy-statement send-static term 2 from route-filter 172.16.7.0/24 exact
user@R4# set policy-statement send-static term 2 then community add R3_PREFERRED
user@R4# set policy-statement send-static term 2 then accept
user@R4# set policy-statement send-static term 3 then reject
```

5. Configure the static routes.

```
[edit routing-options static]
user@R4# set route 172.16.0.0/24 reject
user@R4# set route 172.16.1.0/24 reject
user@R4# set route 172.16.2.0/24 reject
user@R4# set route 172.16.3.0/24 reject
user@R4# set route 172.16.4.0/24 reject
user@R4# set route 172.16.5.0/24 reject
user@R4# set route 172.16.6.0/24 reject
user@R4# set route 172.16.7.0/24 reject
```

6. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R4# set router-id 192.168.0.4
user@R4# set autonomous-system 64511
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R1

```
user@R1# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            address 10.0.0.1/30;
        }
    }
}
ge-0/0/1 {
    unit 0 {
        family inet {
            address 10.1.0.5/30;
        }
    }
```

```
}
ge-0/0/2 {
    unit 0 {
        family inet {
            address 10.0.0.14/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group int {
        type internal;
        local-address 192.168.0.1;
        export send-direct;
        neighbor 192.168.0.2;
        neighbor 192.168.0.3;
    }
    group ext {
        type external;
        import change-local-preference;
        peer-as 64511;
        neighbor 10.0.0.13;
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/0.0;
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
        }
```

```
    }
}
```

```
user@R1# show policy-options
policy-statement change-local-preference {
    term find-R1-routes {
        from community R1_PREFERRED;
        then {
            local-preference 200;
        }
    }
    term find-R3-routes {
        from community R3_PREFERRED;
        then {
            local-preference 50;
        }
    }
}
policy-statement send-direct {
    term 1 {
        from {
            protocol direct;
            route-filter 10.0.0.12/30 exact;
        }
        then accept;
    }
}
community R3_PREFERRED members 64511:3;
community R1_PREFERRED members 64511:1;
```

```
user@R1# show routing-options
router-id 192.168.0.1;
autonomous-system 64510;
```

Device R4

```
user@R4# show interfaces
ge-0/0/0 {
    unit 0 {
```

```
        family inet {
            address 10.0.0.13/30;
        }
    }
}
ge-0/0/1 {
    unit 0 {
        family inet {
            address 10.0.0.9/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.4/32;
        }
    }
}
```

```
user@R4# show protocols
bgp {
    group to-R1 {
        type external;
        export send-static;
        peer-as 64510;
        neighbor 10.0.0.14;
    }
    group to-R3 {
        type external;
        export send-static;
        peer-as 64510;
        neighbor 10.0.0.10;
    }
}
```

```
user@R4# show policy-options
policy-statement send-static {
    term 1 {
        from {
```

```
            protocol static;
            route-filter 172.16.0.0/24 exact;
            route-filter 172.16.1.0/24 exact;
            route-filter 172.16.2.0/24 exact;
            route-filter 172.16.3.0/24 exact;
        }
        then {
            community add R1_PREFERRED;
            accept;
        }
    }
    term 2 {
        from {
            protocol static;
            route-filter 172.16.4.0/24 exact;
            route-filter 172.16.5.0/24 exact;
            route-filter 172.16.6.0/24 exact;
            route-filter 172.16.7.0/24 exact;
        }
        then {
            community add R3_PREFERRED;
            accept;
        }
    }
    term 3 {
        then reject;
    }
}
community R3_PREFERRED members 64511:3;
community R1_PREFERRED members 64511:1;
```

```
user@R4# show routing-options
router-id 192.168.0.4;
autonomous-system 64511;
static {
    route 172.16.0.0/24 reject;
    route 172.16.1.0/24 reject;
    route 172.16.2.0/24 reject;
    route 172.16.3.0/24 reject;
    route 172.16.4.0/24 reject;
    route 172.16.5.0/24 reject;
```

```
    route 172.16.6.0/24 reject;
    route 172.16.7.0/24 reject;
}
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes Sent on Device R4 | **543**
- Verifying the Routes Received on Device R2 | **546**

Confirm that the configuration is working properly.

Verifying the Routes Sent on Device R4

Purpose

On device R4, check the routes sent to device R1 and device R3.

Action

```
user@R4> show route advertising-protocol bgp 10.0.0.14 extensive

inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
* 172.16.0.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.1.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1
```

```
* 172.16.2.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.3.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.4.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.5.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.6.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.7.0/24 (1 entry, 1 announced)
 BGP group to-R1 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3
```

```
user@R4> show route advertising-protocol bgp 10.0.0.10 extensive

inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
```

```
* 172.16.0.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.1.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.2.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.3.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:1

* 172.16.4.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.5.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.6.0/24 (1 entry, 1 announced)
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3

* 172.16.7.0/24 (1 entry, 1 announced)
```

```
 BGP group to-R3 type External
     Nexthop: Self
     AS path: [64511] I
     Communities: 64511:3
```

Meaning

Device R4 has tagged the routes with the communities 64511:1 and 64511:3 and sent them to device R1 and R3.

Verifying the Routes Received on Device R2

Purpose

On device R2, check the routes received from device R1 and device R3.

Action

```
user@R2> show route receive-protocol bgp 192.168.0.1

inet.0: 25 destinations, 25 routes (25 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.12/30            192.168.0.1                  100        I
* 172.16.0.0/24           10.0.0.13                    200        64511 I
* 172.16.1.0/24           10.0.0.13                    200        64511 I
* 172.16.2.0/24           10.0.0.13                    200        64511 I
* 172.16.3.0/24           10.0.0.13                    200        64511 I
```

```
user@R2> show route receive-protocol bgp 192.168.0.3

inet.0: 25 destinations, 25 routes (25 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.0.0.8/30             192.168.0.3                  100        I
* 172.16.4.0/24           10.0.0.9                     200        64511 I
* 172.16.5.0/24           10.0.0.9                     200        64511 I
```

```
* 172.16.6.0/24          10.0.0.9                200        64511 I
* 172.16.7.0/24          10.0.0.9                200        64511 I
```

```
user@R2> show route match-prefix 172.16.*

inet.0: 25 destinations, 25 routes (25 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.0.0/24      *[BGP/170] 1w3d 00:02:11, localpref 200, from 192.168.0.1
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.0.0.1 via ge-0/0/0.0
172.16.1.0/24      *[BGP/170] 1w3d 00:02:11, localpref 200, from 192.168.0.1
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.0.0.1 via ge-0/0/0.0
172.16.2.0/24      *[BGP/170] 1w3d 00:02:11, localpref 200, from 192.168.0.1
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.0.0.1 via ge-0/0/0.0
172.16.3.0/24      *[BGP/170] 1w3d 00:02:11, localpref 200, from 192.168.0.1
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.0.0.1 via ge-0/0/0.0
172.16.4.0/24      *[BGP/170] 1w3d 00:01:50, localpref 200, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.1.0.2 via ge-0/0/1.0
172.16.5.0/24      *[BGP/170] 1w3d 00:01:50, localpref 200, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.1.0.2 via ge-0/0/1.0
172.16.6.0/24      *[BGP/170] 1w3d 00:01:50, localpref 200, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.1.0.2 via ge-0/0/1.0
172.16.7.0/24      *[BGP/170] 1w3d 00:01:50, localpref 200, from 192.168.0.3
                      AS path: 64511 I, validation-state: unverified
                    >  to 10.1.0.2 via ge-0/0/1.0
```

Meaning

Device R2 has the routes with the expected local preferences and the expected active routes, as designated by the asterisks (*).

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

RELATED DOCUMENTATION

Example: Configuring Extended Communities in a Routing Policy | 548

No Link Title

No Link Title

Example: Configuring a Routing Policy to Redistribute BGP Routes with a Specific Community Tag into IS-IS

Example: Configuring Extended Communities in a Routing Policy

IN THIS SECTION

- Requirements | 548
- Overview | 548
- Configuration | 550
- Verification | 555

An extended community is similar in most ways to a regular community. Some networking implementations, such as virtual private networks (VPNs), use extended communities because the 4-octet regular community value does not provide enough expansion and flexibility. An extended community is an eight-octet value divided into two main sections.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 549

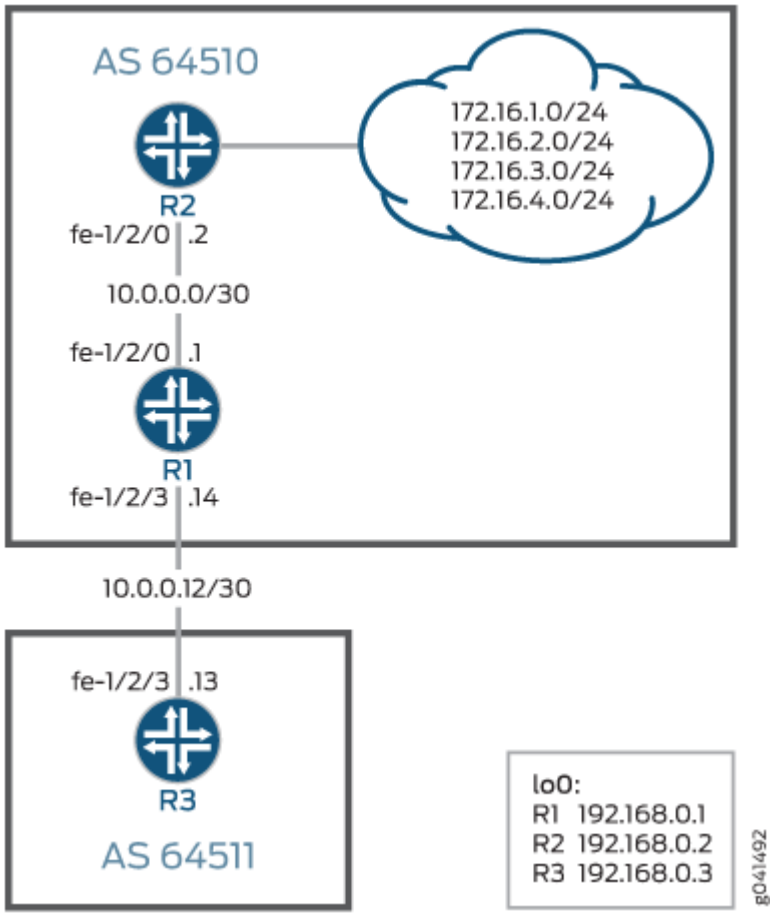
In this example, Device R1 and Device R2 are OSPF neighbors in autonomous system (AS) 64510. Device R3 has an external BGP (EBGP) connection to Device R1. Device R2 has customer networks in

the 172.16/16 address space, simulated with addresses on its loopback interface (lo0). Device R1 has static routes to several 172.16.*x*/24 networks, and attaches regular community values to these routes. Device R1 then uses an export policy to advertise the routes to Device R3. Device R3 receives these routes and uses an import policy to add extended community values to the routes.

Topology

Figure 33 on page 549 shows the sample network.

Figure 33: Topology for Extended BGP Communities



"CLI Quick Configuration" on page 550 shows the configuration for all of the devices in Figure 33 on page 549.

The section "No Link Title" on page 552 describes the steps on Device R3.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 550
- Procedure | 552

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/3 unit 0 family inet address 10.0.0.14/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32 primary
set protocols bgp group ext type external
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.13
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 172.16.1.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.1.0/24 community 64510:1
set routing-options static route 172.16.2.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.2.0/24 community 64510:2
set routing-options static route 172.16.3.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.3.0/24 community 64510:3
set routing-options static route 172.16.4.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.4.0/24 community 64510:4
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces lo0 unit 0 family inet address 172.16.2.2/32
set interfaces lo0 unit 0 family inet address 172.16.3.3/32
set interfaces lo0 unit 0 family inet address 172.16.4.4/32
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64510
```

Device R3

```
set interfaces fe-1/2/3 unit 0 family inet address 10.0.0.13/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group to-R1 type external
set protocols bgp group to-R1 import set-ext-comms
set protocols bgp group to-R1 peer-as 64510
set protocols bgp group to-R1 neighbor 10.0.0.14
set policy-options policy-statement set-ext-comms term route-1 from route-filter 172.16.1.0/24
exact
set policy-options policy-statement set-ext-comms term route-1 then community add target-as
set policy-options policy-statement set-ext-comms term route-1 then accept
set policy-options policy-statement set-ext-comms term route-2 from route-filter 172.16.2.0/24
exact
set policy-options policy-statement set-ext-comms term route-2 then community add target-ip
set policy-options policy-statement set-ext-comms term route-2 then accept
set policy-options policy-statement set-ext-comms term route-3 from route-filter 172.16.3.0/24
exact
set policy-options policy-statement set-ext-comms term route-3 then community add origin-as
set policy-options policy-statement set-ext-comms term route-3 then accept
set policy-options policy-statement set-ext-comms term route-4 from route-filter 172.16.4.0/24
exact
set policy-options policy-statement set-ext-comms term route-4 then community add origin-ip
set policy-options policy-statement set-ext-comms term route-4 then accept
set policy-options community origin-as members origin:64511:3
set policy-options community origin-ip members origin:172.16.7.7:4
set policy-options community target-as members target:64511:1
set policy-options community target-ip members target:172.16.7.7:2
```

```
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64511
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R3:

1. Configure the interfaces.

```
[edit interfaces]
user@R3# set fe-1/2/3 unit 0 family inet address 10.0.0.13/30
user@R3# set lo0 unit 0 family inet address 192.168.0.3/32
```

2. Configure the EBGP connection to Device R1.

```
[edit protocols bgp group to-R1]
user@R3# set type external
user@R3# set import set-ext-comms
user@R3# set peer-as 64510
user@R3# set neighbor 10.0.0.14
```

3. Configure the policy that adds extended community values to the routes received from Device R1.

   An extended community uses a notation of *type*:*administrator*:*assigned-number*.

   The specific community values can be anything that accomplishes your administrative goals, within certain parameters, as explained in *community (Policy Options)*.

```
[edit policy-options policy-statement set-ext-comms]
user@R3# set term route-1 from route-filter 172.16.1.0/24 exact
user@R3# set term route-1 then community add target-as
user@R3# set term route-1 then accept
user@R3# set term route-2 from route-filter 172.16.2.0/24 exact
user@R3# set term route-2 then community add target-ip
user@R3# set term route-2 then accept
```

```
user@R3# set term route-3 from route-filter 172.16.3.0/24 exact
user@R3# set term route-3 then community add origin-as
user@R3# set term route-3 then accept
user@R3# set term route-4 from route-filter 172.16.4.0/24 exact
user@R3# set term route-4 then community add origin-ip
user@R3# set term route-4 then accept
[edit policy-options]
user@R3# set community origin-as members origin:64511:3
user@R3# set community origin-ip members origin:172.16.7.7:4
user@R3# set community target-as members target:64511:1
user@R3# set community target-ip members target:172.16.7.7:2
```

4. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R3# set router-id 192.168.0.3
user@R3# set autonomous-system 64511
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
fe-1/2/3 {
    unit 0 {
        family inet {
            address 10.0.0.13/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.3/32;
        }
```

```
    }
}
```

```
user@R3# show protocols
bgp {
    group to-R1 {
        type external;
        import set-ext-comms;
        peer-as 64510;
        neighbor 10.0.0.14;
    }
}
```

```
user@R3# show policy-options
policy-statement set-ext-comms {
    term route-1 {
        from {
            route-filter 172.16.1.0/24 exact;
        }
        then {
            community add target-as;
            accept;
        }
    }
    term route-2 {
        from {
            route-filter 172.16.2.0/24 exact;
        }
        then {
            community add target-ip;
            accept;
        }
    }
    term route-3 {
        from {
            route-filter 172.16.3.0/24 exact;
        }
        then {
            community add origin-as;
            accept;
```

```
        }
    }
    term route-4 {
        from {
            route-filter 172.16.4.0/24 exact;
        }
        then {
            community add origin-ip;
            accept;
        }
    }
}
community origin-as members origin:64511:3;
community origin-ip members origin:172.16.7.7:4;
community target-as members target:64511:1;
community target-ip members target:172.16.7.7:2;
```

```
user@R3# show routing-options
router-id 192.168.0.3;
autonomous-system 64511;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the Routes on Device R1 | 555
- Verifying the Routes on Device R3 | 557

Confirm that the configuration is working properly.

Verifying the Routes on Device R1

Purpose

On Device R1, check the 172.16. routes in the routing table.

Action

```
user@R1> show route protocol static match-prefix 172.16.* detail

inet.0: 15 destinations, 15 routes (15 active, 0 holddown, 0 hidden)
172.16.1.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 835
                Address: 0x9260250
                Next-hop reference count: 19
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
                State: <Active Int Ext>
                Local AS: 64510
                Age: 2:06:08
                Task: RT
                Announcement bits (2): 2-KRT 3-BGP_RT_Background
                AS path: I
                Communities: 64510:1

172.16.2.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 835
                Address: 0x9260250
                Next-hop reference count: 19
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
                State: <Active Int Ext>
                Local AS: 64510
                Age: 2:06:08
                Task: RT
                Announcement bits (2): 2-KRT 3-BGP_RT_Background
                AS path: I
                Communities: 64510:2

172.16.3.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 835
                Address: 0x9260250
                Next-hop reference count: 19
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
                State: <Active Int Ext>
                Local AS: 64510
                Age: 2:06:08
```

```
                Task: RT
                Announcement bits (2): 2-KRT 3-BGP_RT_Background
                AS path: I
                Communities: 64510:3

172.16.4.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 835
                Address: 0x9260250
                Next-hop reference count: 19
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
                State: <Active Int Ext>
                Local AS: 64510
                Age: 2:06:08
                Task: RT
                Announcement bits (2): 2-KRT 3-BGP_RT_Background
                AS path: I
                Communities: 64510:4
```

Meaning

The output shows that the regular community values are attached to the routes.

NOTE: The communities are attached to static routes, thus demonstrating that communities can be attached to non-BGP routes.

Verifying the Routes on Device R3

Purpose

On Device R3, check the 172.16. routes in the routing table.

Action

```
user@R3> show route protocol bgp match-prefix 172.16.* detail
betsy@tp5# run show route protocol bgp match-prefix 172.16.* detail logical-system R3

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
172.16.1.0/24 (1 entry, 1 announced)
```

```
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 611
                Address: 0x9260130
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                State: <Active Ext>
                Local AS: 64511 Peer AS: 64510
                Age: 1:57:27
                Task: BGP_64510.10.0.0.14+54618
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:1 target:64511:1
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.2.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 611
                Address: 0x9260130
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                State: <Active Ext>
                Local AS: 64511 Peer AS: 64510
                Age: 1:57:27
                Task: BGP_64510.10.0.0.14+54618
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:2 target:172.16.7.7:2
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.3.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 611
                Address: 0x9260130
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                State: <Active Ext>
```

```
                Local AS: 64511 Peer AS: 64510
                Age: 1:57:27
                Task: BGP_64510.10.0.0.14+54618
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:3 origin:64511:3
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.4.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 611
                Address: 0x9260130
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                State: <Active Ext>
                Local AS: 64511 Peer AS: 64510
                Age: 1:57:27
                Task: BGP_64510.10.0.0.14+54618
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:4 origin:172.16.7.7:4
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1
```

Meaning

The output shows that the regular community values remain attached to the routes, and the extended community values are added.

RELATED DOCUMENTATION

Example: Configuring Communities in a Routing Policy | **528**

No Link Title

No Link Title

Example: Configuring a Routing Policy to Redistribute BGP Routes with a Specific Community Tag into IS-IS

Example: Configuring BGP Large Communities

IN THIS SECTION

- Requirements | 560
- Overview | 560
- Configuration | 561
- Verification | 567

This example shows you to configure optional transitive path attribute - a 12-byte BGP large community that provides the most significant 4-byte value to encode autonomous system number as the global administrator and the remaining two 4-byte assigned numbers to encode the local values as defined in RFC 8092. You can configure BGP large community at `[edit policy-options community` *`community-name`* `members]` and `[edit routing-options static route` *`ip-address`* `community]` hierarchy levels. The BGP large community attributes format has four fields: `large:`*`global administrator`*`:`*`assigned number`*`:`*`assigned number`*.

Requirements

This example uses the following hardware and software components:

- Three MX Series routers
- Junos OS Release 17.3 or later running on all devices

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

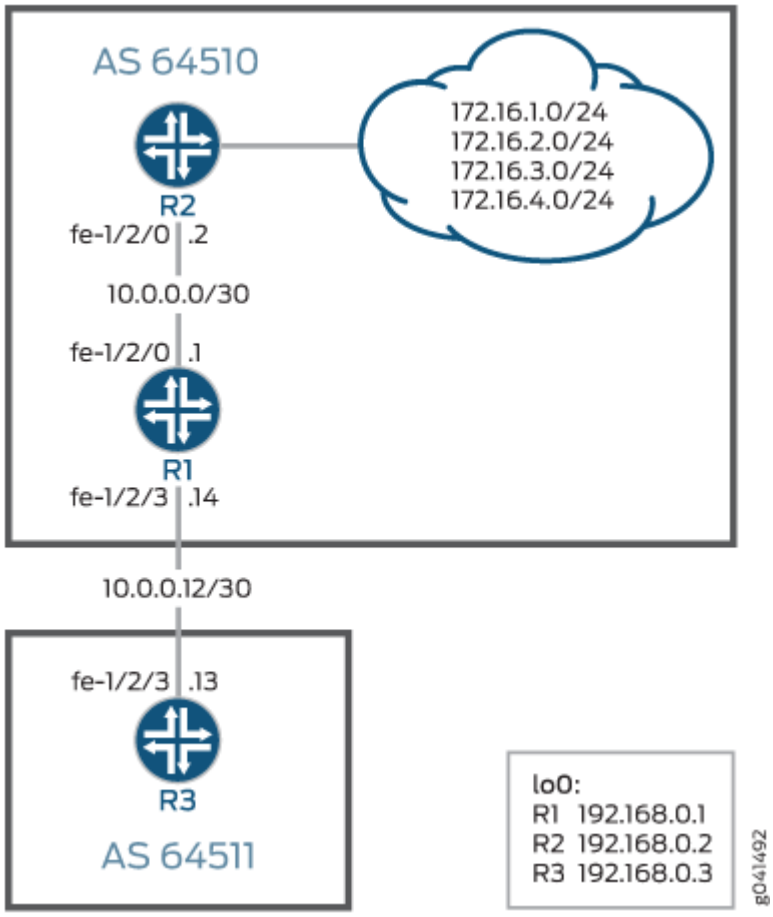
- Topology | 561

In this example, Device R1 and Device R2 are OSPF neighbors in autonomous system (AS) 64510. Device R3 has an external BGP (EBGP) connection to Device R1. Device R2 has customer networks in the 172.16/16 address space, simulated with addresses on its loopback interface (lo0). Device R1 has static routes to several 172.16.x/24 networks, and attaches regular community values to these routes.

Device R1 then uses an export policy to advertise the routes to Device R3. Device R3 receives these routes and uses an import policy to add large community values to the routes.

Topology

Figure 1 shows the sample network.



Configuration

IN THIS SECTION

- CLI Quick Configuration | 562
- Procedure | 563
- Results | 565

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/3 unit 0 family inet address 10.0.0.14/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32 primary
set routing-options static route 172.16.1.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.1.0/24 community 64510:1
set routing-options static route 172.16.1.0/24 community large:64510:100:1
set routing-options static route 172.16.2.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.2.0/24 community 64510:2
set routing-options static route 172.16.2.0/24 community large:64510:200:2
set routing-options static route 172.16.3.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.3.0/24 community 64510:3
set routing-options static route 172.16.4.0/24 next-hop 10.0.0.2
set routing-options static route 172.16.4.0/24 community 64510:4
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
set protocols bgp group ext type external
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.13
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces lo0 unit 0 family inet address 172.16.2.2/32
set interfaces lo0 unit 0 family inet address 172.16.3.3/32
set interfaces lo0 unit 0 family inet address 172.16.4.4/32
set routing-options router-id 192.168.0.2
```

```
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
```

Device R3

```
set interfaces fe-1/2/3 unit 0 family inet address 10.0.0.13/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 64511
set protocols bgp group to-R1 type external
set protocols bgp group to-R1 import set-large-comms
set protocols bgp group to-R1 peer-as 64510
set protocols bgp group to-R1 neighbor 10.0.0.14
set policy-options policy-statement set-large-comms term route-1 from route-filter 172.16.1.0/24
exact
set policy-options policy-statement set-large-comms term route-1 then community add large2-as
set policy-options policy-statement set-large-comms term route-1 then accept
set policy-options policy-statement set-large-comms term route-2 from route-filter 172.16.2.0/24
exact
set policy-options policy-statement set-large-comms term route-2 then community add large2-ip
set policy-options policy-statement set-large-comms term route-2 then accept
set policy-options policy-statement set-large-comms term route-3 from route-filter 172.16.3.0/24
exact
set policy-options policy-statement set-large-comms term route-3 then community add large1-as
set policy-options policy-statement set-large-comms term route-3 then accept
set policy-options policy-statement set-large-comms term route-4 from route-filter 172.16.4.0/24
exact
set policy-options policy-statement set-large-comms term route-4 then community add large1-ip
set policy-options policy-statement set-large-comms term route-4 then accept
set policy-options community large1-as members large:64511:3:1
set policy-options community large1-ip members large:7777:4:1
set policy-options community large2-as members large:64511:1:1
set policy-options community large2-ip members large:7777:2:1
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R3:

1. Configure the interfaces.

```
[edit interfaces]
set fe-1/2/3 unit 0 family inet address 10.0.0.13/30
set lo0 unit 0 family inet address 192.168.0.3/32
```

2. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
set router-id 192.168.0.3
set autonomous-system 64511
```

3. Configure the EBGP connection to Device R1.

```
[edit protocols bgp group to-R1]
set type external
set import set-large-comms
set peer-as 64510
set neighbor 10.0.0.14
```

4. Configure the policy that adds large community values to the routes received from Device R1.

   A large community uses a notation of large:*global administrator*:*assigned number*:*assigned number*. The specific community values can be anything that accomplishes your administrative goals, within certain parameters.

```
[edit policy-options policy-statement set-large-comms]
set term route-1 from route-filter 172.16.1.0/24 exact
set term route-1 then community add large2-as
set term route-1 then accept
set term route-2 from route-filter 172.16.2.0/24 exact
set term route-2 then community add large2-ip
set term route-2 then accept
set term route-3 from route-filter 172.16.3.0/24 exact
set term route-3 then community add large1-as
set term route-3 then accept
set term route-4 from route-filter 172.16.4.0/24 exact
```

```
set term route-4 then community add large1-ip
set term route-4 then accept
```

```
[edit policy-options ]
set community large1-as members large:64511:3:1
set community large1-ip members large:7777:4:1
set community large2-as members large:64511:1:1
set community large2-ip members large:7777:2:1
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
fe-1/2/3 {
    unit 0 {
        family inet {
            address 10.0.0.13/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.3/32;
        }
    }
}
```

```
user@R3# show protocols
bgp {
    group to-R1 {
        type external;
        import set-large-comms;
        peer-as 64510;
        neighbor 10.0.0.14;
```

```
    }
}
```

```
user@R3# show policy-options
policy-statement set-large-comms {
    term route-1 {
        from {
            route-filter 172.16.1.0/24 exact;
        }
        then {
            community add large2-as;
            accept;
        }
    }
    term route-2 {
        from {
            route-filter 172.16.2.0/24 exact;
        }
        then {
            community add large2-ip;
            accept;
        }
    }
    term route-3 {
        from {
            route-filter 172.16.3.0/24 exact;
        }
        then {
            community add large1-as;
            accept;
        }
    }
    term route-4 {
        from {
            route-filter 172.16.4.0/24 exact;
        }
        then {
            community add large1-ip;
            accept;
        }
    }
```

```
  }
community large1-as members large:64511:3:1;
community large1-ip members large:7777:4:1;
community large2-as members large:64511:1:1;
community large2-ip members large:7777:2:1;
```

```
user@R3# show routing-options
router-id 192.168.0.3;
autonomous-system 64511;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying R1 | 567
- Verifying R3 | 569

Confirm that the configuration is working properly.

Verifying R1

Purpose

On Device R1, check the 172.16. routes in the routing table.

Action

```
user@R1> show route protocol static match-prefix 172.16.* detail
inet.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
172.16.1.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 580
                Address: 0xb7a1270
                Next-hop reference count: 9
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
```

```
                Session Id: 0x140
                State: < Active Int Ext >
                Local AS: 64510
                Age: 4d 19:02:23
                Validation State: unverified
                Task: RT
                Announcement bits (2): 0-KRT 4-BGP_RT_Background
                AS path: I
                Communities: 64510:1 large:64510:100:1

172.16.2.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 580
                Address: 0xb7a1270
                Next-hop reference count: 9
                Next hop: 10.0.0.2 via fe-1/2/0.0.0, selected
                Session Id: 0x140
                State: < Active Int Ext >
                Local AS: 64510
                Age: 4d 19:02:23
                Validation State: unverified
                Task: RT
                Announcement bits (2): 0-KRT 4-BGP_RT_Background
                AS path: I
                Communities: 64510:2 large:64510:200:2

172.16.3.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 580
                Address: 0xb7a1270
                Next-hop reference count: 9
                Next hop: 10.0.0.2 via fe-1/2/0.0.0, selected
                Session Id: 0x140
                State: < Active Int Ext >
                Local AS: 64510
                Age: 4d 22:17:12
                Validation State: unverified
                Task: RT
                Announcement bits (2): 0-KRT 4-BGP_RT_Background
                AS path: I
                Communities: 64510:3

172.16.4.0/24 (1 entry, 1 announced)
```

```
        *Static Preference: 5
                Next hop type: Router, Next hop index: 580
                Address: 0xb7a1270
                Next-hop reference count: 9
                Next hop: 10.0.0.2 via fe-1/2/0.0.0, selected
                Session Id: 0x140
                State: < Active Int Ext >
                Local AS: 64510
                Age: 4d 22:17:12
                Validation State: unverified
                Task: RT
                Announcement bits (2): 0-KRT 4-BGP_RT_Background
                AS path: I
                Communities: 64510:4
 . . .
```

Meaning

The output shows that the regular community and large community values are attached to the routes.

NOTE: The communities are attached to static routes, thus demonstrating that both regular and large communities can be attached to static routes.

Verifying R3

Purpose

On Device R3, check the 172.16. routes in the routing table.

Action

```
user@R3> show route protocol bgp match-prefix 172.16.* detail

inet.0: 14 destinations, 14 routes (14 active, 0 holddown, 0 hidden)
172.16.1.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 581
                Address: 0xb7a10f0
                Next-hop reference count: 8
```

```
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                Session Id: 0x140
                State: < Active Ext >
                Local AS: 64511 Peer AS: 64510
                Age: 3d 16:36:18
                Validation State: unverified
                Task: BGP_64510.10.0.0.14
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:1 large:64510:100:1 large:64511:1:1
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.2.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 581
                Address: 0xb7a10f0
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                Session Id: 0x140
                State: < Active Ext >
                Local AS: 64511 Peer AS: 64510
                Age: 3d 16:36:18
                Validation State: unverified
                Task: BGP_64510.10.0.0.14
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:2 large:7777:2:1 large:64510:200:2
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.3.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 581
                Address: 0xb7a10f0
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                Session Id: 0x140
```

```
                State: < Active Ext >
                Local AS: 64511 Peer AS: 64510
                Age: 3d 16:36:18
                Validation State: unverified
                Task: BGP_64510.10.0.0.14
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:3 large:64511:3:1
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1

172.16.4.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 581
                Address: 0xb7a10f0
                Next-hop reference count: 8
                Source: 10.0.0.14
                Next hop: 10.0.0.14 via fe-1/2/3.0, selected
                Session Id: 0x140
                State: < Active Ext >
                Local AS: 64511 Peer AS: 64510
                Age: 3d 16:36:18
                Validation State: unverified
                Task: BGP_64510.10.0.0.14
                Announcement bits (1): 0-KRT
                AS path: 64510 I
                Communities: 64510:4 large:7777:4:1
                Accepted
                Localpref: 100
                Router ID: 192.168.0.1
. . .
```

Meaning

The output shows that the advertised regular and large community values remain attached to the routes, and that the new large community values are added when received by R3.

RELATED DOCUMENTATION

Understanding How to Define BGP Communities and Extended Communities | 513

community

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

IN THIS SECTION

- Requirements | 572
- Overview | 572
- Configuration | 573
- Verification | 580

This example shows how to create a policy that accepts BGP routes based on the number of BGP communities.

Requirements

No special configuration beyond device initialization is required before you configure this example.

Overview

IN THIS SECTION

- Topology | 573

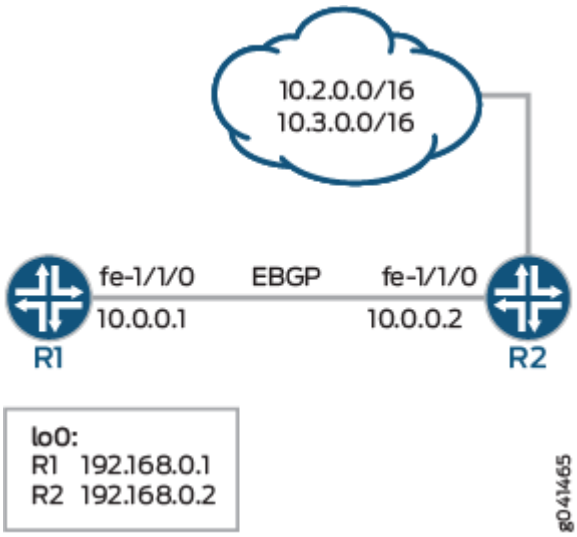
This example shows two routing devices with an external BGP (EBGP) connection between them. Device R2 uses the BGP session to send two static routes to Device R1. On Device R1, an import policy specifies that the BGP-received routes can contain up to five communities to be considered a match. For example, if a route contains three communities, it is considered a match and is accepted. If a route contains six or more communities, it is considered a nonmatch and is rejected.

It is important to remember that the default policy for EBGP is to accept all routes. To ensure that the nonmatching routes are rejected, you must include a `then reject` action at the end of the policy definition.

Topology

Figure 34 on page 573 shows the sample network.

Figure 34: BGP Policy with a Limit on the Number of Communities Accepted



Configuration

IN THIS SECTION

- CLI Quick Configuration | 573
- Procedure | 575

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/1/0 unit 0 description to-R2
set interfaces fe-1/1/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
```

```
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 2
set protocols bgp group external-peers neighbor 10.0.0.2 import import-communities
set policy-options policy-statement import-communities term 1 from protocol bgp
set policy-options policy-statement import-communities term 1 from community-count 5 orlower
set policy-options policy-statement import-communities term 1 then accept
set policy-options policy-statement import-communities term 2 then reject
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 1
```

Device R2

```
set interfaces fe-1/1/0 unit 0 description to-R1
set interfaces fe-1/1/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group external-peers type external
set protocols bgp group external-peers export statics
set protocols bgp group external-peers peer-as 1
set protocols bgp group external-peers neighbor 10.0.0.1
set policy-options policy-statement statics from protocol static
set policy-options policy-statement statics then community add 1
set policy-options policy-statement statics then accept
set policy-options community 1 members 2:1
set policy-options community 1 members 2:2
set policy-options community 1 members 2:3
set policy-options community 1 members 2:4
set policy-options community 1 members 2:5
set policy-options community 1 members 2:6
set policy-options community 1 members 2:7
set policy-options community 1 members 2:8
set policy-options community 1 members 2:9
set policy-options community 1 members 2:10
set routing-options static route 10.2.0.0/16 reject
set routing-options static route 10.2.0.0/16 install
set routing-options static route 10.3.0.0/16 reject
set routing-options static route 10.3.0.0/16 install
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 2
```

Procedure

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the interfaces.

```
[edit interfaces]
user@R1# set fe-1/1/0 unit 0 description to-R2
user@R1# set fe-1/1/0 unit 0 family inet address 10.0.0.1/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure BGP.

   Apply the import policy to the BGP peering session with Device R2.

```
[edit protocols bgp group external-peers]
user@R1# set type external
user@R1# set peer-as 2
user@R1# set neighbor 10.0.0.2 import import-communities
```

3. Configure the routing policy that sends direct routes.

```
[edit policy-options policy-statement import-communities]
user@R1# set term 1 from protocol bgp
user@R1# set term 1 from community-count 5 orlower
user@R1# set term 1 then accept
user@R1# set term 2 then reject
```

4. Configure the autonomous system (AS) number and the router ID.

```
[edit routing-options ]
user@R1# set router-id 192.168.0.1
user@R1# set autonomous-system 1
```

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set fe-1/1/0 unit 0 description to-R1
user@R2# set fe-1/1/0 unit 0 family inet address 10.0.0.2/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the router ID and the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set router-id 192.168.0.3
user@R2# set autonomous-system 2
```

3. Configure BGP.

```
[edit  protocols bgp group external-peers]
user@R2# set type external
user@R2# set peer-as 1
user@R2# set neighbor 10.0.0.1
```

4. Configure multiple communities, or configure a single community with multiple members.

```
[edit policy-options community 1]
user@R2# set members 2:1
user@R2# set members 2:2
user@R2# set members 2:3
user@R2# set members 2:4
user@R2# set members 2:5
user@R2# set members 2:6
user@R2# set members 2:7
user@R2# set members 2:8
```

```
user@R2# set members 2:9
user@R2# set members 2:10
```

5. Configure the static routes.

```
[edit routing-options static]
user@R2# set route 10.2.0.0/16 reject
user@R2# set route 10.2.0.0/16 install
user@R2# set route 10.3.0.0/16 reject
user@R2# set route 10.3.0.0/16 install
```

6. Configure a routing policy that advertises static routes into BGP and adds the BGP community to the routes.

```
[edit policy-options policy-statement statics]
user@R2# set from protocol static
user@R2# set then community add 1
user@R2# set then accept
```

7. Apply the export policy.

```
[edit protocols bgp group external-peers]
user@R2# set export statics
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R1

```
user@R1# show interfaces
fe-1/1/0 {
    unit 0{
        description to-R2;
        family inet {
            address 10.0.0.1/30;
        }
```

```
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
}
```

```
user@R1# show protocols
bgp {
    group external-peers {
        type external;
        peer-as 2;
        neighbor 10.0.0.2 {
            import import-communities;
        }
    }
}
```

```
user@R1# show policy-options
policy-statement import-communities {
    term 1 {
        from {
            protocol bgp;
            community-count 5 orlower;
        }
        then accept;
    }
    term 2 {
        then reject;
```

```
    }
}
```

```
user@R1# show routing-options
router-id 192.168.0.1;
autonomous-system 1;
```

Device R2

```
user@R2# show interfaces
fe-1/1/0 {
    unit 0 {
        description to-R1;
        family inet {
            address 10.0.0.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group external-peers {
        type external;
        export statics;
        peer-as 1;
        neighbor 10.0.0.1;
    }
}
```

```
user@R2# show policy-options
policy-statement statics {
```

```
    from protocol static;
    then {
        community add 1;
        accept;
    }
}
community 1 members [ 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10 ];
```

```
user@R2# show routing-options
static {
    route 10.2.0.0/16 {
        reject;
        install;
    }
    route 10.3.0.0/16 {
        reject;
        install;
    }
}
router-id 192.168.0.3;
autonomous-system 2;
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the BGP Routes | 580

Confirm that the configuration is working properly.

Verifying the BGP Routes

Purpose

Make sure that the routing table on Device R1 contains the expected BGP routes.

Action

1. On Device R1, run the `show route protocols bgp` command.

```
user@R1> show route protocols bgp

inet.0: 5 destinations, 5 routes (3 active, 0 holddown, 2 hidden)
```

2. On Device R1, change the `community-count` configuration in the import policy.

```
[edit policy-options policy-statement import-communities term 1]
user@R1# set from community-count 5 orhigher
user@R1# commit
```

3. On Device R1, run the `show route protocols bgp` command.

```
user@R1> show route protocols bgp

inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.0.0/16        *[BGP/170] 18:29:53, localpref 100
                      AS path: 2 I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/1/0.0
10.3.0.0/16        *[BGP/170] 18:29:53, localpref 100
                      AS path: 2 I, validation-state: unverified
                    > to 10.0.0.2 via fe-1/1/0.0
```

4. On Device R1, run the `show route protocols bgp extensive` command to view the advertised communities.

```
user@R1> show route protocols bgp extensive
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
10.2.0.0/16 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.2.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
```

```
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via fe-1/1/0.0, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 18:56:10
                Validation State: unverified
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Communities: 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3

10.3.0.0/16 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.3.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via fe-1/1/0.0, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 18:56:10
                Validation State: unverified
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Communities: 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3
```

Meaning

The output shows that in Device R1's routing table, the BGP routes sent from Device R2 are hidden. When the `community-count` setting in Device R1's import policy is modified, the BGP routes are no longer hidden.

RELATED DOCUMENTATION

- Example: Configuring a Routing Policy to Redistribute BGP Routes with a Specific Community Tag into IS-IS
- Understanding External BGP Peering Sessions

Example: Configuring a Routing Policy That Removes BGP Communities

IN THIS SECTION

- Requirements | 583
- Overview | 583
- Configuration | 585
- Verification | 592

This example shows how to create a policy that accepts BGP routes, but removes BGP communities from the routes.

Requirements

No special configuration beyond device initialization is required before you configure this example.

Overview

IN THIS SECTION

- Topology | 584

This example shows two routing devices with an external BGP (EBGP) connection between them. Device R2 uses the BGP session to send two static routes to Device R1. On Device R1, an import policy specifies that all BGP communities must be removed from the routes.

By default, when communities are configured on EBGP peers, they are sent and accepted. To suppress the acceptance of communities received from a neighbor, you can remove all communities or a specified set of communities. When the result of a policy is an empty set of communities, the community attribute is not included. To remove all communities, first define a wildcard set of communities (here, the community is named `wild`):

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

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

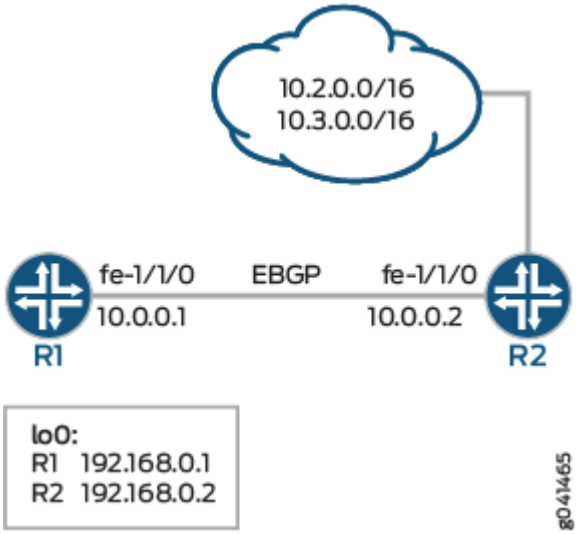
```
[edit policy-options]
policy-statement policy-name {
    term term-name {
        then community delete wild;
    }
}
```

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

Topology

Figure 35 on page 585 shows the sample network.

Figure 35: BGP Policy That Removes Communities



Configuration

IN THIS SECTION

- CLI Quick Configuration | 585
- Procedure | 587

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/1/0 unit 0 description to-R2
set interfaces fe-1/1/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 2
set protocols bgp group external-peers neighbor 10.0.0.2 import remove-communities
set policy-options policy-statement remove-communities term 1 from protocol bgp
set policy-options policy-statement remove-communities term 1 then community delete wild
```

```
set policy-options policy-statement remove-communities term 1 then accept
set policy-options policy-statement remove-communities term 2 then reject
set policy-options community wild members *:*
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 1
```

Device R2

```
set interfaces fe-1/1/0 unit 0 description to-R1
set interfaces fe-1/1/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group external-peers type external
set protocols bgp group external-peers export statics
set protocols bgp group external-peers peer-as 1
set protocols bgp group external-peers neighbor 10.0.0.1
set policy-options policy-statement statics from protocol static
set policy-options policy-statement statics then community add 1
set policy-options policy-statement statics then accept
set policy-options community 1 members 2:1
set policy-options community 1 members 2:2
set policy-options community 1 members 2:3
set policy-options community 1 members 2:4
set policy-options community 1 members 2:5
set policy-options community 1 members 2:6
set policy-options community 1 members 2:7
set policy-options community 1 members 2:8
set policy-options community 1 members 2:9
set policy-options community 1 members 2:10
set routing-options static route 10.2.0.0/16 reject
set routing-options static route 10.2.0.0/16 install
set routing-options static route 10.3.0.0/16 reject
set routing-options static route 10.3.0.0/16 install
set routing-options router-id 192.168.0.3
set routing-options autonomous-system 2
```

Procedure

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the interfaces.

```
[edit interfaces]
user@R1# set fe-1/1/0 unit 0 description to-R2
user@R1# set fe-1/1/0 unit 0 family inet address 10.0.0.1/30
user@R1# set lo0 unit 0 family inet address 192.168.0.1/32
```

2. Configure BGP.

   Apply the import policy to the BGP peering session with Device R2.

```
[edit protocols bgp group external-peers]
user@R1# set type external
user@R1# set peer-as 2
user@R1# set neighbor 10.0.0.2 import remove-communities
```

3. Configure the routing policy that deletes communities.

```
[edit policy-options policy-statement remove-communities]
user@R1# set term 1 from protocol bgp
user@R1# set term 1 then community delete wild
user@R1# set term 1 then accept
user@R1# set term 2 then reject
```

4. Configure the autonomous system (AS) number and the router ID.

```
[edit routing-options ]
user@R1# set router-id 192.168.0.1
user@R1# set autonomous-system 1
```

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set fe-1/1/0 unit 0 description to-R1
user@R2# set fe-1/1/0 unit 0 family inet address 10.0.0.2/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the router ID and the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set router-id 192.168.0.3
user@R2# set autonomous-system 2
```

3. Configure BGP.

```
[edit  protocols bgp group external-peers]
user@R2# set type external
user@R2# set peer-as 1
user@R2# set neighbor 10.0.0.1
```

4. Configure multiple communities, or configure a single community with multiple members.

```
[edit policy-options community 1]
user@R2# set members 2:1
user@R2# set members 2:2
user@R2# set members 2:3
user@R2# set members 2:4
user@R2# set members 2:5
user@R2# set members 2:6
user@R2# set members 2:7
user@R2# set members 2:8
```

```
user@R2# set members 2:9
user@R2# set members 2:10
```

5. Configure the static routes.

```
[edit routing-options static]
user@R2# set route 10.2.0.0/16 reject
user@R2# set route 10.2.0.0/16 install
user@R2# set route 10.3.0.0/16 reject
user@R2# set route 10.3.0.0/16 install
```

6. Configure a routing policy that advertises static routes into BGP and adds the BGP community to the routes.

```
[edit policy-options policy-statement statics]
user@R2# set from protocol static
user@R2# set then community add 1
user@R2# set then accept
```

7. Apply the export policy.

```
[edit protocols bgp group external-peers]
user@R2# set export statics
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R1

```
user@R1# show interfaces
fe-1/1/0 {
    unit 0{
        description to-R2;
        family inet {
            address 10.0.0.1/30;
        }
```

```
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group external-peers {
        type external;
        peer-as 2;
        neighbor 10.0.0.2 {
            import remove-communities;
        }
    }
}
```

```
user@R1# show policy-options
policy-statement remove-communities {
    term 1 {
        from protocol bgp;
        then {
            community delete wild;
            accept;
        }
    }
    term 2 {
        then reject;
    }
```

```
}
community wild members *:*;
```

```
user@R1# show routing-options
router-id 192.168.0.1;
autonomous-system 1;
```

Device R2

```
user@R2# show interfaces
fe-1/1/0 {
    unit 0 {
        description to-R1;
        family inet {
            address 10.0.0.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group external-peers {
        type external;
        export statics;
        peer-as 1;
        neighbor 10.0.0.1;
    }
}
```

```
user@R2# show policy-options
policy-statement statics {
```

```
    from protocol static;
    then {
        community add 1;
        accept;
    }
}
community 1 members [ 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10 ];
```

```
user@R2# show routing-options
static {
    route 10.2.0.0/16 {
        reject;
        install;
    }
    route 10.3.0.0/16 {
        reject;
        install;
    }
}
router-id 192.168.0.3;
autonomous-system 2;
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the BGP Routes | 592

Confirm that the configuration is working properly.

Verifying the BGP Routes

Purpose

Make sure that the routing table on Device R1 does not contain BGP communities.

Action

1. On Device R1, run the `show route protocols bgp extensive` command.

```
user@R1> show route protocols bgp extensive

inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
10.2.0.0/16 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.2.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via lt-1/1/0.5, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 20:39:01
                Validation State: unverified
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3

10.3.0.0/16 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.3.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via lt-1/1/0.5, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 20:39:01
                Validation State: unverified
```

```
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3
```

2. On Device R1, deactivate the `community remove` configuration in the import policy.

```
[edit policy-options policy-statement remove-communities term 1]
user@R1# deactivate then community delete wild
user@R1# commit
```

3. On Device R1, run the `show route protocols bgp extensive` command to view the advertised communities.

```
user@R1> show route protocols bgp extensive
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
10.2.0.0/16 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.2.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via lt-1/1/0.5, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 20:40:53
                Validation State: unverified
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Communities: 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3

10.3.0.0/16 (1 entry, 1 announced)
```

```
TSI:
KRT in-kernel 10.3.0.0/16 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 671
                Address: 0x9458270
                Next-hop reference count: 4
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via lt-1/1/0.5, selected
                Session Id: 0x100001
                State: <Active Ext>
                Local AS:     1 Peer AS:     2
                Age: 20:40:53
                Validation State: unverified
                Task: BGP_2.10.0.0.2+179
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Communities: 2:1 2:2 2:3 2:4 2:5 2:6 2:7 2:8 2:9 2:10
                Accepted
                Localpref: 100
                Router ID: 192.168.0.3
```

Meaning

The output shows that in Device R1's routing table, the communities are suppressed in the BGP routes sent from Device R2. When the `community remove` setting in Device R1's import policy is deactivated, the communities are no longer suppressed.

RELATED DOCUMENTATION

- Example: Configuring a Routing Policy to Redistribute BGP Routes with a Specific Community Tag into IS-IS
- Understanding External BGP Peering Sessions

CHAPTER 7

Increasing Network Stability with BGP Route Flapping Actions

IN THIS CHAPTER

- Understanding Damping Parameters | **596**
- Using Routing Policies to Damp BGP Route Flapping | **598**
- Example: Configuring BGP Route Flap Damping Parameters | **604**
- Example: Configuring BGP Route Flap Damping Based on the MBGP MVPN Address Family | **619**

Understanding Damping Parameters

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 method of reducing the number of update messages sent between BGP peers, thereby reducing the load on these peers, without adversely affecting the route convergence time for stable routes.

Flap damping reduces the number of update messages by marking routes as ineligible for selection as the active or preferable route. Marking routes in this way 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 (routes in different ASs). You can also apply flap damping within a confederation, between confederation member ASs. Because routing consistency within an AS is important, do not apply flap damping to internal BGP (IBGP) routes. (If you do, it is ignored.)

There is an exception that rule. Starting in Junos OS Release 12.2, you can apply flap damping at the address family level. In a Junos OS Release 12.2 or later installation, when you apply flap damping at the address family level, it works for both IBGP and EBGP.

By default, route flap damping is not enabled. Damping is applied to external peers and to peers at confederation boundaries.

When you enable damping, default parameters are applied, as summarized in Table 23 on page 597.

Table 23: Damping Parameters

Damping Parameter	Description	Default Value	Possible Values
half-life ***minutes***	Decay half-life—Number of minutes after which an arbitrary value is halved if a route stays stable.	**15** (minutes)	**1** through **45**
max-suppress ***minutes***	Maximum hold-down time for a route, in minutes.	**60** (minutes)	**1** through **720**
reuse	Reuse threshold—Arbitrary value below which a suppressed route can be used again.	**750**	**1** through **20,000**
suppress	Cutoff (suppression) threshold—Arbitrary value above which a route can no longer be used or included in advertisements.	**3000**	**1** through **20,000**

To change 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. For the damping routing policy to work, you also must enable BGP route flap damping.

Release History Table

Release	Description
12.2	Starting in Junos OS Release 12.2, you can apply flap damping at the address family level.

RELATED DOCUMENTATION

Understanding Routing Policies

Using Routing Policies to Damp BGP Route Flapping

IN THIS SECTION

- Configuring BGP Flap Damping Parameters | 598
- Specifying BGP Flap Damping as the Action in Routing Policy Terms | 601
- Disabling Damping for Specific Address Prefixes | 602
- Configuring BGP Flap Damping | 602

BGP *route flapping* describes the situation in which BGP systems send an excessive number of update messages to advertise network reachability information. BGP *flap damping* is a way to reduce the number of update messages sent between BGP peers, thereby reducing the load on these peers without adversely affecting the route convergence time.

Flap damping reduces the number of update messages by marking routes as ineligible for selection as the active or preferable route. Doing this leads to some delay, or *suppression*, in the propagation of route information, but the result is increased network stability. You typically apply flap damping to external BGP (EBGP) routes (that is, to routes in different ASs). You can also apply it within a confederation, between confederation member ASs. Because routing consistency within an AS is important, do not apply flap damping to IBGP routes. (If you do, it is ignored.)

BGP flap damping is defined in RFC 2439, *BGP Route Flap Damping.*

To effect changes to the default BGP flap damping values, you define actions by creating a named set of damping parameters and including it in a routing policy with the `damping` action (described in "Configuring Actions That Manipulate Route Characteristics" on page 77). For the damping routing policy to work, you also must enable BGP route flap damping.

The following sections discuss the following topics:

Configuring BGP Flap Damping Parameters

To define damping parameters, include the `damping` statement:

```
[edit policy-options]
damping name {
    disable;
    half-life minutes;
    max-suppress minutes;
```

```
    reuse number;
    suppress number;
}
```

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 24 on page 599.

Table 24: Damping Parameters

Damping Parameter	Description	Default	Possible Values
`half-life` *minutes*	Decay half-life, in minutes	15 minutes	1 through 45 minutes
`max-suppress` *minutes*	Maximum hold-down time, in minutes	60 minutes	1 through 720 minutes
`reuse`	Reuse threshold	750 (unitless)	1 through 20,000 (unitless)
`suppress`	Cutoff (suppression) threshold	3000 (unitless)	1 through 20,000 (unitless)

If you do not specify one or more of the damping parameters, the default value of the parameter is used.

To understand how to configure these parameters, you need to understand how damping suppresses routes. How long a route can be suppressed is based on a *figure of merit*, which is a value that correlates to the probability of future instability of a route. Routes with higher figure-of-merit values are suppressed for longer periods of time. The figure-of-merit value decays exponentially over time.

A figure-of-merit value of zero is assigned to each new route. The value is increased each time the route is withdrawn or readvertised, or when one of its path attributes changes. With each incident of instability, the value increases as follows:

- Route is withdrawn—1000
- Route is readvertised—1000
- Route’s path attributes change—500

NOTE: Other vendors' implementations for figure-of-merit increase the value only when a route is withdrawn. The Junos OS implementation for figure-of-merit increases the value for both route withdrawal and route readvertisement. To accommodate other implementations for figure-of-merit, multiply the `reuse` and `suppress` threshold values by 2.

When a route's figure-of-merit value reaches a particular level, called the *cutoff* or *suppression threshold*, the route is suppressed. If a route is suppressed, the routing table no longer installs the route into the forwarding table and no longer exports this route to any of the routing protocols. By default, a route is suppressed when its figure-of-merit value reaches 3000. To modify this default, include the `suppress` option at the `[edit policy-options damping` *`name`*`]` hierarchy level.

If a route has flapped, but then becomes stable so that none of the incidents listed previously occur within a configurable amount of time, the figure-of-merit value for the route decays exponentially. The default half-life is 15 minutes. For example, for a route with a figure-of-merit value of 1500, if no incidents occur, its figure-of-merit value is reduced to 750 after 15 minutes and to 375 after another 15 minutes. To modify the default half-life, include the `half-life` option at the `[edit policy-options damping` *`name`*`]` hierarchy level.

NOTE: For the half-life, configure a value that is less than the max-suppress. If you do not, the configuration is rejected.

A suppressed route becomes reusable when its figure-of-merit value decays to a value below a *reuse threshold*, thus allowing routes that experience transient instability to once again be considered valid. The default reuse threshold is 750. When the figure-of-merit value passes below the reuse threshold, the route once again is considered usable and can be installed in the forwarding table and exported from the routing table. To modify the default reuse threshold, include the `reuse` option at the `[edit policy-options damping` *`name`*`]` hierarchy level.

The maximum suppression time provides an upper bound on the time that a route can remain suppressed. The default maximum suppression time is 60 minutes. To modify the default, include the `max-suppress` option at the `[edit policy-options damping` *`name`*`]` hierarchy level.

NOTE: For the max-suppress, configure a value that is greater than the half-life. If you do not, the configuration is rejected.

A route's figure-of-merit value stops increasing when it reaches a maximum suppression threshold, which is determined based on the route's suppression threshold level, half-life, reuse threshold, and maximum hold-down time.

The merit ceiling, ε_c, which is the maximum merit that a flapping route can collect, is calculated using the following formula:

$$\varepsilon_c \leq \varepsilon_r \, e^{(t/\lambda)(\ln 2)}$$

ε_r is the figure-of-merit reuse threshold, t is the maximum hold-down time in minutes, and λ is the half-life in minutes. For example, if you use the default figure-of-merit values in this formula, but use a half-life of 30 minutes, the calculation is as follows:

$$\varepsilon_c \leq 750 \, e^{(120/30)(\ln 2)}$$

$$\varepsilon_c \leq 12000$$

NOTE: The cutoff threshold, which you configure using the `suppress` option, must be less than or equal to the merit ceiling, ε_c. If the configured cutoff threshold or the default cutoff threshold is greater than the merit ceiling, the route is never suppressed and damping never occurs.

To display figure-of-merit information, use the `show policy damping` command.

A route that has been assigned a figure of merit is considered to have a damping state. To display the current damping information on the routing device, use the `show route detail` command.

Specifying BGP Flap Damping as the Action in Routing Policy Terms

To BGP flap damping as the action in a routing policy term, include the `damping` statement and the name of the configured damping parameters either as an option of the `route-filter` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `from]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name from]
route-filter destination-prefix match-type {
    damping damping-parameters;
}
```

or at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
damping damping-parameters;
```

Disabling Damping for Specific Address Prefixes

Normally, you enable or disable damping on a per-peer basis. However, you can disable damping for a specific prefix received from a peer by including the `disable` option:

```
[edit policy-options damping name]
disable;
```

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;
    }
}
```

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
```

RELATED DOCUMENTATION

Example: Configuring BGP Route Flap Damping Parameters | 604

Example: Configuring BGP Route Flap Damping Based on the MBGP MVPN Address Family | 619

Example: Configuring BGP Route Flap Damping Parameters

IN THIS SECTION

- Requirements | 605
- Overview | 605
- Configuration | 606

Verification | 612

This example shows how to configure damping parameters.

Requirements

Before you begin, configure router interfaces and configure routing protocols.

Overview

This example has three routing devices. Device R2 has external BGP (EBGP) connections with Device R1 and Device R3.

Device R1 and Device R3 have some static routes configured for testing purposes, and these static routes are advertised through BGP to Device R2.

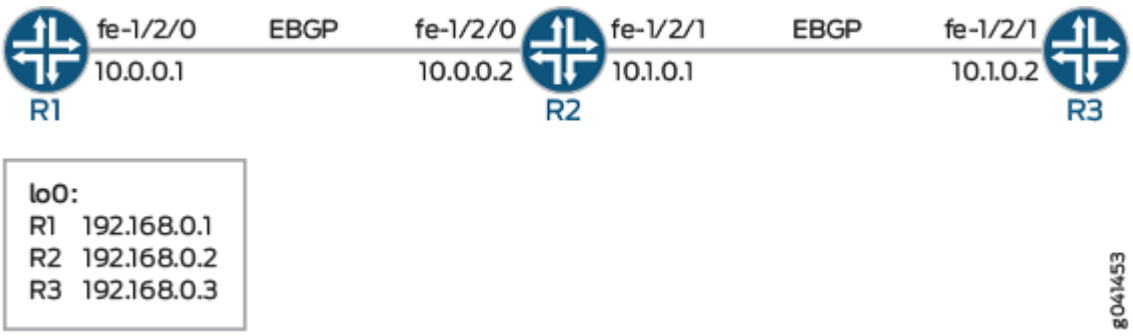
Device R2 damps routes received from Device R1 and Device R3 according to these criteria:

- Damp all prefixes with a mask length equal to or greater than 17 more aggressively than routes with a mask length between 9 and 16.
- Damp routes with a mask length between 0 and 8, inclusive, less than routes with a mask length greater than 8.
- Do not damp the 10.128.0.0/9 prefix at all.

The routing policy is evaluated when routes are being exported from the routing table into the forwarding table. Only the active routes are exported from the routing table.

Figure 36 on page 605 shows the sample network.

Figure 36: BGP Flap Damping Topology



"CLI Quick Configuration" on page 606 shows the configuration for all of the devices in Figure 36 on page 605.

The section "No Link Title" on page 608 describes the steps on Device R2.

Configuration

IN THIS SECTION

- Procedure | 606

Procedure

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct-and-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-direct-and-static term 1 from protocol direct
set policy-options policy-statement send-direct-and-static term 1 from protocol static
set policy-options policy-statement send-direct-and-static term 1 then accept
set routing-options static route 172.16.0.0/16 reject
set routing-options static route 172.16.128.0/17 reject
set routing-options static route 172.16.192.0/20 reject
set routing-options static route 10.0.0.0/9 reject
set routing-options static route 172.16.233.0/7 reject
set routing-options static route 10.224.0.0/11 reject
set routing-options static route 0.0.0.0/0 reject
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp damping
set protocols bgp group ext type external
set protocols bgp group ext import damp
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
set policy-options policy-statement damp term 1 from route-filter 10.128.0.0/9 exact damping dry
set policy-options policy-statement damp term 1 from route-filter 0.0.0.0/0 prefix-length-
range /0-/8 damping timid
set policy-options policy-statement damp term 1 from route-filter 0.0.0.0/0 prefix-length-
range /17-/32 damping aggressive
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options damping aggressive half-life 30
set policy-options damping aggressive suppress 2500
set policy-options damping timid half-life 5
set policy-options damping dry disable
set routing-options autonomous-system 200
```

Device R3

```
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct-and-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.1.0.1
set policy-options policy-statement send-direct-and-static term 1 from protocol direct
set policy-options policy-statement send-direct-and-static term 1 from protocol static
set policy-options policy-statement send-direct-and-static term 1 then accept
set routing-options static route 10.128.0.0/9 reject
set routing-options autonomous-system 300
```

Step-by-Step Procedure

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

To configure damping parameters:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set fe-1/2/1 unit 0 family inet address 10.1.0.1/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure the BGP neighbors.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set neighbor 10.0.0.1 peer-as 100
user@R2# set neighbor 10.1.0.2 peer-as 300
```

3. Create and configure the damping parameter groups.

```
[edit policy-options]
user@R2# set damping aggressive half-life 30
user@R2# set damping aggressive suppress 2500
user@R2# set damping timid half-life 5
user@R2# set damping dry disable
```

4. Configure the damping policy.

```
[edit policy-options policy-statement damp term 1]
user@R2# set from route-filter 10.128.0.0/9 exact damping dry
user@R2# set from route-filter 0.0.0.0/0 prefix-length-range /0-/8 damping timid
user@R2# set from route-filter 0.0.0.0/0 prefix-length-range /17-/32 damping aggressive
```

5. Enable damping for BGP.

```
[edit protocols bgp]
user@R2# set damping
```

6. Apply the policy as an import policy for the BGP neighbor.

```
[edit protocols bgp group ext]
user@R2# set import damp
```

> **NOTE**: You can refer to the same routing policy one or more times in the same or different `import` statements.

7. Configure an export policy.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

8. Apply the export policy.

```
[edit protocols bgp group ext]
user@R2# set export send-direct
```

9. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set autonomous-system 200
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 10.1.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    damping;
    group ext {
        type external;
        import damp;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
        neighbor 10.1.0.2 {
            peer-as 300;
```

```
        }
    }
}
```

```
user@R2# show policy-options
policy-statement damp {
    term 1 {
        from {
            route-filter 10.128.0.0/9 exact damping dry;
            route-filter 0.0.0.0/0 prefix-length-range /0-/8 damping timid;
            route-filter 0.0.0.0/0 prefix-length-range /17-/32 damping aggressive;
        }
    }
}
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
damping aggressive {
    half-life 30;
    suppress 2500;
}
damping timid {
    half-life 5;
}
damping dry {
    disable;
}
```

```
user@R2# show routing-options
autonomous-system 200;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Causing Some Routes to Flap | 612
- Checking the Route Flaps | 613
- Verifying Route Flap Damping | 614
- Displaying the Details of a Damped Route | 615
- Verifying That Default Damping Parameters Are in Effect | 616
- Filtering the Damping Information | 617

Confirm that the configuration is working properly.

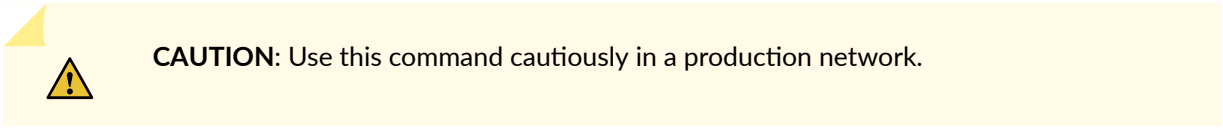
Causing Some Routes to Flap

Purpose

To verify your route flap damping policy, some routes must flap. Having a live Internet feed almost guarantees that a certain number of route flaps will be present. If you have control over a remote system that is advertising the routes, you can modify the advertising router's policy to effect the advertisement and withdrawal of all routes or of a given prefix. In a test environment, you can cause routes to flap by clearing the BGP neighbors or by restarting the routing process on the BGP neighbors, as shown here.

Action

From operational mode on Device R1 and Device R3, enter the `restart routing` command.




```
user@R1> restart routing

R1 started, pid 10474
```

```
user@R3> restart routing

R3 started, pid 10478
```

Meaning

On Device R2, all of the routes from the neighbors are withdrawn and re-advertised.

Checking the Route Flaps

Purpose

View the number of neighbor flaps.

Action

From operational mode, enter the `show bgp summary` command.

```
user@R2> show bgp summary

Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                      12          1         11          0          11          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.0.0.1                100         10         10       0       4        2:50
0/9/0/9              0/0/0/0
```

```
10.1.0.2               300         10         10       0       4        2:53
1/3/1/2             0/0/0/0
```

Meaning

This output was captured after the routing process was restarted on Device R2's neighbors four times.

Verifying Route Flap Damping

Purpose

Verify that routes are being hidden due to damping.

Action

From operational mode, enter the `show route damping suppressed` command.

```
user@R2> show route damping suppressed

inet.0: 15 destinations, 17 routes (6 active, 0 holddown, 11 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0           [BGP ] 00:00:12, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
10.0.0.0/9          [BGP ] 00:00:12, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
10.0.0.0/30         [BGP ] 00:00:12, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
10.1.0.0/30         [BGP ] 00:00:15, localpref 100
                      AS path: 300 I, validation-state: unverified
                    > to 10.1.0.2 via fe-1/2/1.0
10.224.0.0/11       [BGP ] 00:00:12, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.0.0/16       [BGP ] 00:00:12, localpref 100
                      AS path: 100 I, validation-state: unverified
                    > to 10.0.0.1 via fe-1/2/0.0
172.16.128.0/17     [BGP ] 00:00:12, localpref 100
```

```
                    AS path: 100 I, validation-state: unverified
                  > to 10.0.0.1 via fe-1/2/0.0
172.16.192.0/20   [BGP ] 00:00:12, localpref 100
                    AS path: 100 I, validation-state: unverified
                  > to 10.0.0.1 via fe-1/2/0.0
192.168.0.1/32    [BGP ] 00:00:12, localpref 100
                    AS path: 100 I, validation-state: unverified
                  > to 10.0.0.1 via fe-1/2/0.0
192.168.0.3/32    [BGP ] 00:00:15, localpref 100
                    AS path: 300 I, validation-state: unverified
                  > to 10.1.0.2 via fe-1/2/1.0
172.16.233.0/7      [BGP ] 00:00:12, localpref 100
                    AS path: 100 I, validation-state: unverified
                  > to 10.0.0.1 via fe-1/2/0.0
```

Meaning

The output shows some routing instability. Eleven routes are hidden due to damping.

Displaying the Details of a Damped Route

Purpose

Display the details of damped routes.

Action

From operational mode, enter the `show route damping suppressed 172.16.192.0/20 detail` command.

```
user@R2> show route damping suppressed 172.16.192.0/20 detail

inet.0: 15 destinations, 17 routes (6 active, 0 holddown, 11 hidden)
172.16.192.0/20 (1 entry, 0 announced)
         BGP                 /-101
                Next hop type: Router, Next hop index: 758
                Address: 0x9414484
                Next-hop reference count: 9
                Source: 10.0.0.1
                Next hop: 10.0.0.1 via fe-1/2/0.0, selected
                Session Id: 0x100201
                State: <Hidden Ext>
```

```
                Local AS:   200 Peer AS:   100
                Age: 52
                Validation State: unverified
                Task: BGP_100.10.0.0.1+55922
                AS path: 100 I
                Localpref: 100
                Router ID: 192.168.0.1
                Merit (last update/now): 4278/4196
                damping-parameters: aggressive
                Last update:       00:00:52 First update:       01:01:55
                Flaps: 8
                Suppressed. Reusable in:       01:14:40
                Preference will be: 170
```

Meaning

This output indicates that the displayed route has a mask length that is equal to or greater than /17, and confirms that it has been correctly mapped to the aggressive damping profile. You can also see the route's current (and last) figure of merit value, and when the route is expected to become active if it remains stable.

Verifying That Default Damping Parameters Are in Effect

Purpose

Locating a damped route with a /16 mask confirms that the default parameters are in effect.

Action

From operational mode, enter the `show route damping suppressed detail | match 0/16` command.

```
user@R2> show route damping suppressed detail | match 0/16

172.16.0.0/16 (1 entry, 0 announced)
```

```
user@R2> show route damping suppressed 172.16.0.0/16 detail

inet.0: 15 destinations, 17 routes (6 active, 0 holddown, 11 hidden)
172.16.0.0/16 (1 entry, 0 announced)
```

```
BGP                /-101
        Next hop type: Router, Next hop index: 758
        Address: 0x9414484
        Next-hop reference count: 9
        Source: 10.0.0.1
        Next hop: 10.0.0.1 via fe-1/2/0.0, selected
        Session Id: 0x100201
        State: <Hidden Ext>
        Local AS:   200 Peer AS:   100
        Age: 1:58
        Validation State: unverified
        Task: BGP_100.10.0.0.1+55922
        AS path: 100 I
        Localpref: 100
        Router ID: 192.168.0.1
        Merit (last update/now): 3486/3202
        Default damping parameters used
        Last update:       00:01:58 First update:       01:03:01
        Flaps: 8
        Suppressed. Reusable in:       00:31:40
        Preference will be: 170
```

Meaning

Routes with a /16 mask are not impacted by the custom damping rules. Therefore, the default damping rules are in effect.

To repeat, the custom rules are as follows:

- Damp all prefixes with a mask length equal to or greater than 17 more aggressively than routes with a mask length between 9 and 16.
- Damp routes with a mask length between 0 and 8, inclusive, less than routes with a mask length greater than 8.
- Do not damp the 10.128.0.0/9 prefix at all.

Filtering the Damping Information

Purpose

Use OR groupings or cascaded piping to simplify the determination of what damping profile is being used for routes with a given mask length.

Action

From operational mode, enter the `show route damping suppressed` command.

```
user@R2> show route damping suppressed detail | match "0 announced | damp"

0.0.0.0/0 (1 entry, 0 announced)
                damping-parameters: timid
10.0.0.0/9 (1 entry, 0 announced)
                Default damping parameters used
                damping-parameters: aggressive
                damping-parameters: aggressive
10.224.0.0/11 (1 entry, 0 announced)
                Default damping parameters used
172.16.0.0/16 (1 entry, 0 announced)
                Default damping parameters used
172.16.128.0/17 (1 entry, 0 announced)
                damping-parameters: aggressive
172.16.192.0/20 (1 entry, 0 announced)
                damping-parameters: aggressive
192.168.0.1/32 (1 entry, 0 announced)
                damping-parameters: aggressive
192.168.0.3/32 (1 entry, 0 announced)
                damping-parameters: aggressive
172.16.233.0/7 (1 entry, 0 announced)
                damping-parameters: timid
```

Meaning

When you are satisfied that your EBGP routes are correctly associated with a damping profile, you can issue the `clear bgp damping` operational mode command to restore an active status to your damped routes, which will return your connectivity to normal operation.

RELATED DOCUMENTATION

Understanding Damping Parameters

Using Routing Policies to Damp BGP Route Flapping | 598

Example: Configuring BGP Route Flap Damping Based on the MBGP MVPN Address Family

IN THIS SECTION

- Requirements | 619
- Overview | 619
- Configuration | 620
- Verification | 632

This example shows how to configure an multiprotocol BGP multicast VPN (also called Next-Generation MVPN) with BGP route flap damping.

Requirements

This example uses Junos OS Release 12.2. BGP route flap damping support for MBGP MVPN, specifically, and on an address family basis, in general, is introduced in Junos OS Release 12.2.

Overview

IN THIS SECTION

- Topology | 619

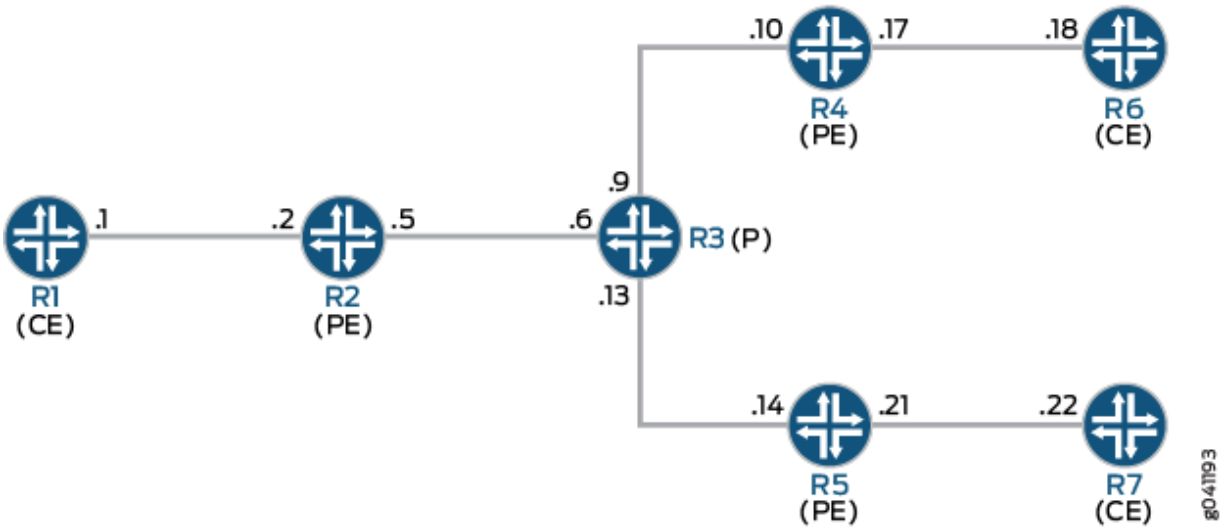
BGP route flap damping helps to diminish route instability caused by routes being repeatedly withdrawn and readvertised when a link is intermittently failing.

This example uses the default damping parameters and demonstrates an MBGP MVPN scenario with three provider edge (PE) routing devices, three customer edge (CE) routing devices, and one provider (P) routing device.

Topology

Figure 37 on page 620 shows the topology used in this example.

Figure 37: MBGP MVPN with BGP Route Flap Damping



On PE Device R4, BGP route flap damping is configured for address family `inet-mvpn`. A routing policy called `dampPolicy` uses the `nlri-route-type` match condition to damp only MVPN route types 3, 4, and 5. All other MVPN route types are not damped.

This example shows the full configuration on all devices in the "CLI Quick Configuration" on page 620 section. The "Configuring Device R4" on page 625 section shows the step-by-step configuration for PE Device R4.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **620**
- Configuring Device R4 | **625**
- Results | **628**

CLI Quick Configuration

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

Device R1

```
set interfaces ge-1/2/0 unit 1 family inet address 10.1.1.1/30
set interfaces ge-1/2/0 unit 1 family mpls
set interfaces lo0 unit 1 family inet address 172.16.1.1/32
set protocols ospf area 0.0.0.0 interface lo0.1 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.1
set protocols pim rp static address 172.16.100.1
set protocols pim interface all
set routing-options router-id 172.16.1.1
```

Device R2

```
set interfaces ge-1/2/0 unit 2 family inet address 10.1.1.2/30
set interfaces ge-1/2/0 unit 2 family mpls
set interfaces ge-1/2/1 unit 5 family inet address 10.1.1.5/30
set interfaces ge-1/2/1 unit 5 family mpls
set interfaces vt-1/2/0 unit 2 family inet
set interfaces lo0 unit 2 family inet address 172.16.1.2/32
set interfaces lo0 unit 102 family inet address 172.16.100.1/32
set protocols mpls interface ge-1/2/1.5
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 172.16.1.2
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp family inet-mvpn signaling
set protocols bgp group ibgp neighbor 172.16.1.4
set protocols bgp group ibgp neighbor 172.16.1.5
set protocols ospf area 0.0.0.0 interface lo0.2 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/1.5
set protocols ldp interface ge-1/2/1.5
set protocols ldp p2mp
set policy-options policy-statement parent_vpn_routes from protocol bgp
set policy-options policy-statement parent_vpn_routes then accept
set routing-instances vpn-1 instance-type vrf
set routing-instances vpn-1 interface ge-1/2/0.2
set routing-instances vpn-1 interface vt-1/2/0.2
set routing-instances vpn-1 interface lo0.102
set routing-instances vpn-1 route-distinguisher 100:100
set routing-instances vpn-1 provider-tunnel ldp-p2mp
set routing-instances vpn-1 vrf-target target:1:1
set routing-instances vpn-1 protocols ospf export parent_vpn_routes
```

```
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface lo0.102 passive
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface ge-1/2/0.2
set routing-instances vpn-1 protocols pim rp static address 172.16.1.2 with 172.16.4.1100.1
set routing-instances vpn-1 protocols pim interface ge-1/2/0.2 mode sparse
set routing-instances vpn-1 protocols mvpn
set routing-options router-id 172.16.1.2
set routing-options autonomous-system 1001
```

Device R3

```
set interfaces ge-1/2/0 unit 6 family inet address 10.1.1.6/30
set interfaces ge-1/2/0 unit 6 family mpls
set interfaces ge-1/2/1 unit 9 family inet address 10.1.1.9/30
set interfaces ge-1/2/1 unit 9 family mpls
set interfaces ge-1/2/2 unit 13 family inet address 10.1.1.13/30
set interfaces ge-1/2/2 unit 13 family mpls
set interfaces lo0 unit 3 family inet address 172.16.1.3/32
set protocols mpls interface ge-1/2/0.6
set protocols mpls interface ge-1/2/1.9
set protocols mpls interface ge-1/2/2.13
set protocols ospf area 0.0.0.0 interface lo0.3 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.6
set protocols ospf area 0.0.0.0 interface ge-1/2/1.9
set protocols ospf area 0.0.0.0 interface ge-1/2/2.13
set protocols ldp interface ge-1/2/0.6
set protocols ldp interface ge-1/2/1.9
set protocols ldp interface ge-1/2/2.13
set protocols ldp p2mp
set routing-options router-id 172.16.1.3
```

Device R4

```
set interfaces ge-1/2/0 unit 10 family inet address 10.1.1.10/30
set interfaces ge-1/2/0 unit 10 family mpls
set interfaces ge-1/2/1 unit 17 family inet address 10.1.1.17/30
set interfaces ge-1/2/1 unit 17 family mpls
set interfaces vt-1/2/0 unit 4 family inet
set interfaces lo0 unit 4 family inet address 172.16.1.4/32
set interfaces lo0 unit 104 family inet address 172.16.100.1/32
set protocols rsvp interface all aggregate
set protocols mpls interface all
```

```
set protocols mpls interface ge-1/2/0.10
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 172.16.1.4
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp family inet-mvpn signaling damping
set protocols bgp group ibgp neighbor 172.16.1.2 import dampPolicy
set protocols bgp group ibgp neighbor 172.16.1.5
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.4 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.10
set protocols ldp interface ge-1/2/0.10
set protocols ldp p2mp
set policy-options policy-statement dampPolicy term term1 from family inet-mvpn
set policy-options policy-statement dampPolicy term term1 from nlri-route-type 3
set policy-options policy-statement dampPolicy term term1 from nlri-route-type 4
set policy-options policy-statement dampPolicy term term1 from nlri-route-type 5
set policy-options policy-statement dampPolicy term term1 then accept
set policy-options policy-statement dampPolicy then damping no-damp
set policy-options policy-statement dampPolicy then accept
set policy-options policy-statement parent_vpn_routes from protocol bgp
set policy-options policy-statement parent_vpn_routes then accept
set policy-options damping no-damp disable
set routing-instances vpn-1 instance-type vrf
set routing-instances vpn-1 interface vt-1/2/0.4
set routing-instances vpn-1 interface ge-1/2/1.17
set routing-instances vpn-1 interface lo0.104
set routing-instances vpn-1 route-distinguisher 100:100
set routing-instances vpn-1 vrf-target target:1:1
set routing-instances vpn-1 protocols ospf export parent_vpn_routes
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface lo0.104 passive
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface ge-1/2/1.17
set routing-instances vpn-1 protocols pim rp static address 172.16.100.1
set routing-instances vpn-1 protocols pim interface ge-1/2/1.17 mode sparse
set routing-instances vpn-1 protocols mvpn
set routing-options router-id 172.16.1.4
set routing-options autonomous-system 64501
```

Device R5

```
set interfaces ge-1/2/0 unit 14 family inet address 10.1.1.14/30
set interfaces ge-1/2/0 unit 14 family mpls
set interfaces ge-1/2/1 unit 21 family inet address 10.1.1.21/30
set interfaces ge-1/2/1 unit 21 family mpls
set interfaces vt-1/2/0 unit 5 family inet
set interfaces lo0 unit 5 family inet address 172.16.1.5/32
set interfaces lo0 unit 105 family inet address 172.16.100.5/32
set protocols mpls interface ge-1/2/0.14
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 172.16.1.5
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp family inet-mvpn signaling
set protocols bgp group ibgp neighbor 172.16.1.2
set protocols bgp group ibgp neighbor 172.16.1.4
set protocols ospf area 0.0.0.0 interface lo0.5 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.14
set protocols ldp interface ge-1/2/0.14
set protocols ldp p2mp
set policy-options policy-statement parent_vpn_routes from protocol bgp
set policy-options policy-statement parent_vpn_routes then accept
set routing-instances vpn-1 instance-type vrf
set routing-instances vpn-1 interface vt-1/2/0.5
set routing-instances vpn-1 interface ge-1/2/1.21
set routing-instances vpn-1 interface lo0.105
set routing-instances vpn-1 route-distinguisher 100:100
set routing-instances vpn-1 vrf-target target:1:1
set routing-instances vpn-1 protocols ospf export parent_vpn_routes
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface lo0.105 passive
set routing-instances vpn-1 protocols ospf area 0.0.0.0 interface ge-1/2/1.21
set routing-instances vpn-1 protocols pim rp static address 172.16.100.2
set routing-instances vpn-1 protocols pim interface ge-1/2/1.21 mode sparse
set routing-instances vpn-1 protocols mvpn
set routing-options router-id 172.16.1.5
set routing-options autonomous-system 1001
```

Device R6

```
set interfaces ge-1/2/0 unit 18 family inet address 10.1.1.18/30
set interfaces ge-1/2/0 unit 18 family mpls
```

```
set interfaces lo0 unit 6 family inet address 172.16.1.6/32
set protocols sap listen 233.1.1.1
set protocols ospf area 0.0.0.0 interface lo0.6 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.18
set protocols pim rp static address 172.16.100.2
set protocols pim interface all
set routing-options router-id 172.16.1.6
```

Device R7

```
set interfaces ge-1/2/0 unit 22 family inet address 10.1.1.22/30
set interfaces ge-1/2/0 unit 22 family mpls
set interfaces lo0 unit 7 family inet address 172.16.1.7/32
set protocols ospf area 0.0.0.0 interface lo0.7 passive
set protocols ospf area 0.0.0.0 interface ge-1/2/0.22
set protocols pim rp static address 172.16.100.2
set protocols pim interface all
set routing-options router-id 172.16.1.7
```

Configuring Device R4

Step-by-Step Procedure

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

To configure Device R4:

1. Configure the interfaces.

```
[edit interfaces]
user@R4# set ge-1/2/0 unit 10 family inet address 10.1.1.10/30
user@R4# set ge-1/2/0 unit 10 family mpls
user@R4# set ge-1/2/1 unit 17 family inet address 10.1.1.17/30
user@R4# set ge-1/2/1 unit 17 family mpls
user@R4# set vt-1/2/0 unit 4 family inet
user@R4# set lo0 unit 4 family inet address 172.16.1.4/32
user@R4# set lo0 unit 104 family inet address 172.16.100.4/32
```

2. Configure MPLS and the signaling protocols on the interfaces.

```
[edit protocols]
user@R4# set mpls interface all
user@R4# set mpls interface ge-1/2/0.10
user@R4# set rsvp interface all aggregate
user@R4# set ldp interface ge-1/2/0.10
user@R4# set ldp p2mp
```

3. Configure BGP.

   The BGP configuration enables BGP route flap damping for the `inet-mvpn` address family. The BGP configuration also imports into the routing table the routing policy called `dampPolicy`. This policy is applied to neighbor PE Device R2.

```
[edit protocols bgp group ibgp]
user@R4# set type internal
user@R4# set local-address 172.16.1.4
user@R4# set family inet-vpn unicast
user@R4# set family inet-vpn any
user@R4# set family inet-mvpn signaling damping
user@R4# set neighbor 172.16.1.2 import dampPolicy
user@R4# set neighbor 172.16.1.5
```

4. Configure an interior gateway protocol.

```
[edit protocols ospf]
user@R4# set traffic-engineering
[edit protocols ospf area 0.0.0.0]
user@R4# set interface all
user@R4# set interface lo0.4 passive
user@R4# set interface ge-1/2/0.10
```

5. Configure a damping policy that uses the `nlri-route-type` match condition to damp only MVPN route types 3, 4, and 5.

```
[edit policy-options policy-statement dampPolicy term term1]
user@R4# set from family inet-mvpn
user@R4# set from nlri-route-type 3
```

```
user@R4# set from nlri-route-type 4
user@R4# set from nlri-route-type 5
user@R4# set then accept
```

6. Configure the `damping` policy to disable BGP route flap damping.

   The `no-damp` policy (`damping no-damp disable`) causes any damping state that is present in the routing table to be deleted. The `then damping no-damp` statement applies the `no-damp` policy as an action and has no `from` match conditions. Therefore, all routes that are not matched by `term1` are matched by this term, with the result that all other MVPN route types are not damped.

```
[edit policy-options policy-statement dampPolicy]
user@R4# set then damping no-damp
user@R4# set then accept
[edit policy-options]
user@R4# set damping no-damp disable
```

7. Configure the `parent_vpn_routes` to accept all other BGP routes that are not from the `inet-mvpn` address family.

   This policy is applied as an OSPF export policy in the routing instance.

```
[edit policy-options policy-statement parent_vpn_routes]
user@R4# set from protocol bgp
user@R4# set then accept
```

8. Configure the VPN routing and forwarding (VRF) instance.

```
[edit routing-instances vpn-1]
user@R4# set instance-type vrf
user@R4# set interface vt-1/2/0.4
user@R4# set interface ge-1/2/1.17
user@R4# set interface lo0.104
user@R4# set route-distinguisher 100:100
user@R4# set vrf-target target:1:1
user@R4# set protocols ospf export parent_vpn_routes
user@R4# set protocols ospf area 0.0.0.0 interface lo0.104 passive
user@R4# set protocols ospf area 0.0.0.0 interface ge-1/2/1.17
user@R4# set protocols pim rp static address 172.16.100.2
```

```
user@R4# set protocols pim interface ge-1/2/1.17 mode sparse
user@R4# set protocols mvpn
```

9. Configure the router ID and the autonomous system (AS) number.

```
[edit routing-options]
user@R4# set router-id 172.16.1.4
user@R4# set autonomous-system 1001
```

10. If you are done configuring the device, commit the configuration.

```
user@R4# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, `show routing-instances`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R4# show interfaces
ge-1/2/0 {
    unit 10 {
        family inet {
            address 10.1.1.10/30;
        }
        family mpls;
    }
}
ge-1/2/1 {
    unit 17 {
        family inet {
            address 10.1.1.17/30;
        }
        family mpls;
    }
}
vt-1/2/0 {
    unit 4 {
        family inet;
```

```
    }
}
lo0 {
    unit 4 {
        family inet {
            address 172.16.1.4/32;
        }
    }
    unit 104 {
        family inet {
            address 172.16.100.4/32;
        }
    }
}
```

```
user@R4# show protocols
rsvp {
    interface all {
        aggregate;
    }
}
mpls {
    interface all;
    interface ge-1/2/0.10;
}
bgp {
    group ibgp {
        type internal;
        local-address 172.16.1.4;
        family inet-vpn {
            unicast;
            any;
        }
        family inet-mvpn {
            signaling {
                damping;
            }
        }
        neighbor 172.16.1.2 {
            import dampPolicy;
        }
```

```
        neighbor 172.16.1.5;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface lo0.4 {
            passive;
        }
        interface ge-1/2/0.10;
    }
}
ldp {
    interface ge-1/2/0.10;
    p2mp;
}
```

```
user@R4# show policy-options
policy-statement dampPolicy {
    term term1 {
        from {
            family inet-mvpn;
            nlri-route-type [ 3 4 5 ];
        }
        then accept;
    }
    then {
        damping no-damp;
        accept;
    }
}
policy-statement parent_vpn_routes {
    from protocol bgp;
    then accept;
}
damping no-damp {
```

```
    disable;
}
```

```
user@R4# show routing-instances
vpn-1 {
    instance-type vrf;
    interface vt-1/2/0.4;
    interface ge-1/2/1.17;
    interface lo0.104;
    route-distinguisher 100:100;
    vrf-target target:1:1;
    protocols {
        ospf {
            export parent_vpn_routes;
            area 0.0.0.0 {
                interface lo0.104 {
                    passive;
                }
                interface ge-1/2/1.17;
            }
        }
        pim {
            rp {
                static {
                    address 172.16.100.2;
                }
            }
            interface ge-1/2/1.17 {
                mode sparse;
            }
        }
        mvpn;
    }
}
```

```
user@R4# show routing-optons
router-id 172.16.1.4;
autonomous-system 1001;
```

Verification

IN THIS SECTION

- Verifying That Route Flap Damping Is Disabled | 632
- Verifying Route Flap Damping | 633

Confirm that the configuration is working properly.

Verifying That Route Flap Damping Is Disabled

Purpose

Verify the presence of the `no-damp` policy, which disables damping for MVPN route types other than 3, 4, and 5.

Action

From operational mode, enter the `show policy damping` command.

```
user@R4> show policy damping
Default damping information:
  Halflife: 15 minutes
  Reuse merit: 750 Suppress/cutoff merit: 3000
  Maximum suppress time: 60 minutes
  Computed values:
    Merit ceiling: 12110
    Maximum decay: 6193
Damping information for "no-damp":
  Damping disabled
```

Meaning

The output shows that the default damping parameters are in effect and that the `no-damp` policy is also in effect for the specified route types.

Verifying Route Flap Damping

Purpose

Check whether BGP routes have been damped.

Action

From operational mode, enter the `show bgp summary` command.

```
user@R4> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0
                       6          6          0          0          0          0
bgp.l3vpn.2
                       0          0          0          0          0          0
bgp.mvpn.0
                       2          2          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.1.2               1001       3159       3155       0       0    23:43:47 Establ
  bgp.l3vpn.0: 3/3/3/0
  bgp.l3vpn.2: 0/0/0/0
  bgp.mvpn.0: 1/1/1/0
  vpn-1.inet.0: 3/3/3/0
  vpn-1.mvpn.0: 1/1/1/0
172.16.1.5               1001       3157       3154       0       0    23:43:40 Establ
  bgp.l3vpn.0: 3/3/3/0
  bgp.l3vpn.2: 0/0/0/0
  bgp.mvpn.0: 1/1/1/0
  vpn-1.inet.0: 3/3/3/0
  vpn-1.mvpn.0: 1/1/1/0
```

Meaning

The Damp State field shows that zero routes in the bgp.mvpn.0 routing table have been damped. Further down, the last number in the State field shows that zero routes have been damped for BGP peer 172.16.1.2.

RELATED DOCUMENTATION

Understanding Damping Parameters

Using Routing Policies to Damp BGP Route Flapping | **598**

Example: Configuring BGP Route Flap Damping Parameters

CHAPTER 8

Tracking Traffic Usage with Source Class Usage and Destination Class Usage Actions

IN THIS CHAPTER

- Understanding Source Class Usage and Destination Class Usage Options | **635**
- Source Class Usage Overview | **637**
- Guidelines for Configuring SCU | **638**
- System Requirements for SCU | **639**
- Terms and Acronyms for SCU | **640**
- Roadmap for Configuring SCU | **641**
- Roadmap for Configuring SCU with Layer 3 VPNs | **642**
- Configuring Route Filters and Source Classes in a Routing Policy | **642**
- Applying the Policy to the Forwarding Table | **644**
- Enabling Accounting on Inbound and Outbound Interfaces | **644**
- Configuring Input SCU on the vt Interface of the Egress PE Router | **645**
- Mapping the SCU-Enabled vt Interface to the VRF Instance | **646**
- Configuring SCU on the Output Interface | **647**
- Associating an Accounting Profile with SCU Classes | **648**
- Verifying Your SCU Accounting Profile | **649**
- SCU Configuration | **650**
- SCU with Layer 3 VPNs Configuration | **661**
- Example: Grouping Source and Destination Prefixes into a Forwarding Class | **671**

Understanding Source Class Usage and Destination Class Usage Options

You can maintain packet counts based on the entry and exit points for traffic passing through your network. Entry and exit points are identified by source and destination prefixes grouped into disjoint

sets defined as source classes and *destination classes*. You can define classes based on a variety of parameters, such as routing neighbors, autonomous systems, and route filters.

Source class usage (SCU) counts packets sent to customers by performing lookups on the IP source address and the IP destination address. SCU makes it possible to track traffic originating from specific prefixes on the provider core and destined for specific prefixes on the customer edge. You must enable SCU accounting on both the inbound and outbound physical interfaces.

Destination class usage (DCU) counts packets from customers by performing lookups of the IP destination address. DCU makes it possible to track traffic originating from the customer edge and destined for specific prefixes on the provider core router.

On T Series Core Routers and M320 Multiservice Edge Routers, the source class and destination classes are not carried across the platform fabric. The implications of this are as follows:

- On T Series and M320 routers, SCU and DCU accounting is performed before the packet enters the fabric.
- On T Series and M320 routers, DCU is performed before output filters are evaluated.
- On M Series platforms, DCU is performed after output filters are evaluated.
- If an output filter drops traffic on M Series devices, the dropped packets are excluded from DCU statistics.
- If an output filter drops traffic on T Series and M320 routers, the dropped packets are included in DCU statistics.

NOTE: For PTX Series routers with FPC3, and PTX1000 routers, to support SCU and DCU, you must configure enhanced-mode on the chassis.

On MX Series platforms with MPC/MIC interfaces, SCU and DCU are performed after output filters are evaluated. Packets dropped by output filters are not included in SCU or DCU statistics.

On MX Series platforms with non-MPC/MIC interfaces, SCU and DCU are performed before output filters are evaluated. Packets dropped by output filters are included in SCU and DCU statistics.

On PTX Series platforms, SCU and DCU accounting is performed before output filters are evaluated. Packets dropped by output filters are included in SCU and DCU statistics. On PTX10003, PTX10004, PTX10008, PTX10001-36MR, and line card JNP10K-LC1201, the systems prefixes with SCU and DCU classes assigned occupy more space in the forwarding information base (FIB) tables than regular routes. You must limit the number of prefixes with non-default class assigned.

On Enhanced Scaling FPCs (T640-FPC1-ES, T640-FPC2-ES, T640-FPC3-ES, T640-FPC4-1P-ES , and T1600-FPC4-ES), the source class accounting is performed at ingress. Starting with Junos OS Release

14.2, the SCU accounting is performed at ingress on a T4000 Type 5 FPC. The implications of this are as follows:

- SCU accounting is performed when packets traverse from T4000 Type 5 FPC (ingress FPC) to Enhanced Scaling FPCs (egress FPC).
- SCU accounting is performed when packets traverse from Enhanced Scaling FPCs (ingress FPC) to T4000 Type 5 FPC (egress FPC).

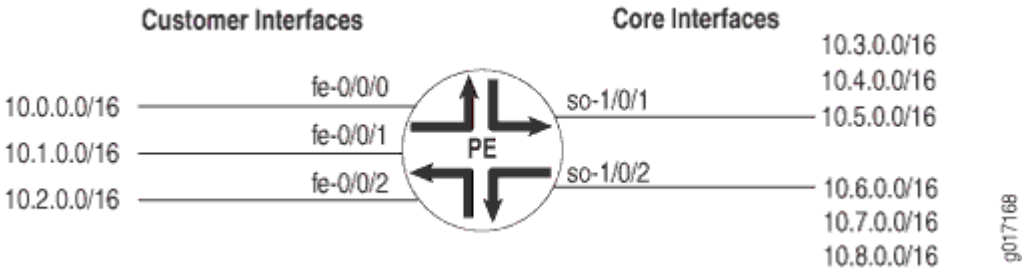
NOTE: When the interface statistics are cleared and then the routing engine is replaced, the SCU and DCU statistics will not match the statistics of the previous routing engine.

For more information about source class usage, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide and the Junos OS Network Interfaces Library for Routing Devices.

Source Class Usage Overview

Source class usage (SCU) is a logical extension of the destination class usage (DCU) concept. DCU was created so that Juniper Networks customers could count on a per-interface basis how much traffic was sent to specified prefixes. Figure 38 on page 637 shows a service provider edge (PE) router diagram.

Figure 38: DCU/SCU Concept



The Fast Ethernet interfaces contain inbound traffic from customers, and the SONET/SDH interfaces are connected to outbound public network prefixes. With DCU configured on the Fast Ethernet interfaces, 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 (in this case, the Fast Ethernet 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. For example, DCU can process cumulative

traffic headed toward interface `fe-0/0/0`, but cannot differentiate between traffic coming only from `10.3.0.0/16` and traffic coming from all prefixes.

You can track source-based traffic by using SCU, which allows 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.

RELATED DOCUMENTATION

- System Requirements for SCU | **639**
- Roadmap for Configuring SCU | **641**
- Roadmap for Configuring SCU with Layer 3 VPNs | **642**
- SCU Configuration | **650**
- SCU with Layer 3 VPNs Configuration | **661**

Guidelines for Configuring SCU

When you enable SCU or DCU, keep the following information in mind:

- In Junos OS Release 5.6 and later for M Series routers only, you can use a source class or a destination class as a match condition in a *firewall filter*. To configure, include the **destination-class** or `source-class` statement at the `[edit firewall filter` *`firewall-name`* `term` *`term-name`* `from]` hierarchy level. For more information about firewall filters, see the *Junos Policy Framework Configuration Guide*.
- You can assign up to 126 source classes and 126 destination classes.
- When configuring policy action statements, you can configure only one source class for each matching route. In other words, more than one source class cannot be applied to the same route.
- A source or destination class is applied to a packet only once during the routing table lookup. When a network prefix matches a class-usage policy, SCU is assigned to packets first; DCU is assigned only if SCU has not been assigned. Be careful when using both class types, since misconfiguration can result in uncounted packets. The following example explores one potential mishap:

  A packet arrives on a router interface configured for both SCU and DCU. The packet's source address matches an SCU class, and its destination matches a DCU class. Consequently, the packet is subjected to a source lookup and is marked with the SCU class. The DCU class is ignored. As a result, the packet is forwarded to the outbound interface with only the SCU class still intact.

  However, the outbound interface lacks an SCU configuration. When the packet is ready to leave the router, the router detects that the output interface is not configured for SCU and the packet is not

counted by SCU. Likewise, even though the prefix matched the DCU prefix, the DCU counters do not increment because DCU was superseded by SCU at the inbound interface.

To solve this problem, make sure you configure both the inbound and outbound interfaces completely or configure only one class type per interface per direction.

- Classes cannot be mapped to directly connected prefixes configured on local interfaces. This is true for DCU and SCU classes.
- If you use multiple terms within a single policy, you only need to configure the policy name and apply it to the forwarding table once. This makes it easier to change options within your terms without having to reconfigure the main policy.
- Execute command line interface (CLI) `show` commands and accounting profiles at the desired outbound interface to track SCU traffic. SCU counters increment at the SCU **output** interface.
- Apply your classes to the inbound and outbound interfaces by means of the **input** and **output** SCU interface parameters.
- On M320 and T Series routers, the source and destination classes are not carried across the platform fabric. For these routers, SCU and DCU accounting is performed before the packet enters the fabric and DCU is performed before output filters are evaluated.
- If an output filter drops traffic on M Series routers other than the M120 router and M320 router, the dropped packets are excluded from DCU statistics. If an output filter drops traffic on M320 and T Series routers, the dropped packets are included in DCU statistics.

RELATED DOCUMENTATION

Source Class Usage Overview | **637**

System Requirements for SCU | **639**

Roadmap for Configuring SCU | **641**

SCU Configuration | **650**

System Requirements for SCU

To implement SCU, your system must meet these requirements:

- Junos OS Release 8.2 or later for M120 and MX Series router support
- Junos OS Release 6.2 or later for IPv6 SCU

- Junos OS Release 5.6 or later to use a source class or a destination class as a match condition in a firewall filter
- Junos OS Release 5.4 or later for IPv4 SCU
- Three Juniper Networks M Series, MX Series, or T Series routers for basic SCU and five routers for SCU with Layer 3 VPNs. One router acts as a source class usage transit router, and the other routers are used to generate traffic or participate in the Layer 3 VPN.
- For M Series and T Series routers, a Tunnel Services PIC for SCU with Layer 3 VPNs

RELATED DOCUMENTATION

Source Class Usage Overview | **637**
Roadmap for Configuring SCU | **641**
Roadmap for Configuring SCU with Layer 3 VPNs | **642**
SCU Configuration | **650**
SCU with Layer 3 VPNs Configuration | **661**

Terms and Acronyms for SCU

IN THIS SECTION

destination class usage (DCU) | **640**
source class usage (SCU) | **641**
source address (SA) | **641**
destination address (DA) | **641**

destination class usage (DCU)

A method of grouping certain types of traffic and monitoring these groups through CLI `show` commands, accounting profiles, or SNMP. DCU uses a destination address lookup when determining group membership. For more information about DCU, see the *Junos Policy Framework Configuration Guide*.

source class usage (SCU)

A method of grouping certain types of traffic and monitoring these groups through CLI `show` commands, accounting profiles, or SNMP. SCU uses a source address lookup when determining group membership. For more information about SCU, see the *Junos Policy Framework Configuration Guide*.

source address (SA)

The IP address of a device sending a packet. This address is included in the IP header and is analyzed by the router for a variety of services, including source-based filtering, policing, class of service (CoS), and SCU.

destination address (DA)

The IP address of a device intended as the receiver for a packet. This address is included in the IP header and is the main address analyzed by the router during routing table lookups and DCU.

Roadmap for Configuring SCU

To configure source class usage (SCU), you must:

1. Create a routing policy that includes prefix route filters that indicate the IPv4 or IPv6 source addresses to monitor. See "Configuring Route Filters and Source Classes in a Routing Policy" on page 642.
2. Apply the filters to the forwarding table. See "Applying the Policy to the Forwarding Table" on page 644.
3. Enable accounting on the inbound and outbound interfaces. See "Enabling Accounting on Inbound and Outbound Interfaces" on page 644.

RELATED DOCUMENTATION

Source Class Usage Overview | 637

System Requirements for SCU | 639

SCU Configuration | 650

Roadmap for Configuring SCU with Layer 3 VPNs

SCU can be implemented over regular interfaces; it is also used in combination with Layer 3 VPNs. When you view SCU traffic on an ingress provider edge (PE) router, use the standard procedure outlined in "Roadmap for Configuring SCU" on page 641. However, when you enable packet counting for Layer 3 VPNs at the egress point of the MPLS tunnel, you need to take some additional steps, as follows:

1. Configure SCU on the virtual loopback tunnel (vt) interface of the egress PE router. See "Configuring Input SCU on the vt Interface of the Egress PE Router" on page 645.
2. Map the SCU-enabled input interface of that router to the virtual routing and forwarding (VRF) instance. See "Mapping the SCU-Enabled vt Interface to the VRF Instance" on page 646.
3. Configure SCU on the output interface of the egress router. See "Configuring SCU on the Output Interface" on page 647.
4. Configure an accounting profile and associate the source class with that accounting profile. You can also specify the filename for the data capture, a class usage profile name, and an interval indicating how often you want the SCU information to be saved. See "Associating an Accounting Profile with SCU Classes" on page 648.

NOTE: SCU is not supported over Layer 2 VPNs.

RELATED DOCUMENTATION

Source Class Usage Overview | 637

System Requirements for SCU | 639

SCU with Layer 3 VPNs Configuration | 661

Configuring Route Filters and Source Classes in a Routing Policy

Begin configuring SCU by creating prefix route filters in a policy statement. These prefixes indicate the IPv4 or IPv6 source addresses to monitor. Within the policy statement, you must define and name the source classes attached to the filters.

```
[edit policy-options]
policy-statement policy-name {
```

```
    term term-name {
        from {
            route-filter address/prefix;
        }
        then source-class class-name;
    }
}
```

NOTE: When configuring policy action statements, you can configure only one source class for each matching route. In other words, more than one source class cannot be applied to the same route.

An alternate configuration method, using the **forwarding-class** policy action, is even more flexible. It allows your IPv4 or IPv6 route filters to apply to an SCU profile, a DCU profile, or both simultaneously. Additionally, if you have only one term, you can implement the **from** and `then` statements at the `[edit policy-options policy-statement` *policy-name*`]` hierarchy level.

```
[edit policy-options]
policy-statement policy-name {
    from {
        route-filter 105.15.0.0/16 orlonger;
    }
    then forwarding-class class-name;
}
```

A third option is the existing DCU parameter of **destination-class**. For more information on DCU, see the *Junos Policy Framework Configuration Guide*.

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU | 641
- SCU Configuration | 650

Applying the Policy to the Forwarding Table

Next, apply the policy you created to the forwarding table. When you apply the policy, the network prefixes you defined are marked with the appropriate source class.

```
[edit routing-options]
forwarding-table {
    export policy-name;
}
```

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU | 641
- SCU Configuration | 650

Enabling Accounting on Inbound and Outbound Interfaces

Unlike DCU, which only requires implementation on a single interface, accounting for SCU must be enabled on two interfaces: the inbound and outbound physical or logical interfaces traversed by the source class. You must define explicitly the two interfaces on which SCU monitored traffic is expected to arrive and depart. This is because SCU performs two lookups in the routing table: a source address (SA) and a destination address (DA) lookup. In contrast, DCU only has a single destination address lookup. By specifying the addresses involved in the additional SCU SA lookup, you minimize the performance impact on your router.

An individual SCU interface can be configured as an input interface, an output interface, or both. SCU can be enabled in an IPv4 (**family inet**) or IPv6 (**family inet6**) network. To configure SCU accounting, include the `source-class-usage` statement at the `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family (inet | inet 6) accounting]` hierarchy level:

```
[edit]
interfaces {
    interface-name {
        unit unit-number {
            family (inet | inet6) {
```

```
                accounting {
                    source-class-usage {
                        (input | output | input output);
                    }
                    destination-class-usage;
                }
            }
        }
    }
}
```

After the full SCU configuration is enabled, every packet arriving on an SCU input interface is subjected to an SA-based lookup and then a DA-based lookup. In addition, an individual set of counters for every configured SCU class is maintained by the router on a per-interface and per-protocol family basis.

RELATED DOCUMENTATION

- Source Class Usage Overview | **637**
- System Requirements for SCU | **639**
- Roadmap for Configuring SCU | **641**
- SCU Configuration | **650**

Configuring Input SCU on the vt Interface of the Egress PE Router

To enable SCU in a Layer 3 VPN, configure source class usage on the virtual loopback tunnel (**vt**) interface of the egress PE router that is either configured for or equipped with a Tunnel PIC. The interface is equivalent to the inbound SCU interface, so use the `input` statement at the `[edit interfaces` *`vt-interface-number`* `unit 0 family inet accounting source-class-usage]` hierarchy level:

```
[edit]
interfaces {
    vt-0/3/0 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
                        input;
                    }
```

```
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU with Layer 3 VPNs | 642
- SCU with Layer 3 VPNs Configuration | 661

Mapping the SCU-Enabled vt Interface to the VRF Instance

Next, include the VPN loopback tunnel interface in the desired VRF instance at the `[edit routing-instances routing-instance-name]` hierarchy level:

```
[edit]
routing-instances {
                                    routing-instance-name                {
        instance-type vrf;
        interface at-2/1/1.0;
        interface vt-0/3/0.0;
        route-distinguisher 10.250.14.225:100;
        vrf-import import-policy-name;
        vrf-export export-policy-name;
        protocols {
            bgp {
                group to-r4 {
                    local-address 10.20.253.1;
                    peer-as 400;
                    neighbor 10.20.253.2;
                }
            }
        }
```

```
    }
}
```

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU with Layer 3 VPNs | 642
- SCU with Layer 3 VPNs Configuration | 661

Configuring SCU on the Output Interface

Since VPN traffic enters the egress router through the VPN loopback tunnel interface, you still need to determine the exit interface for this traffic. To complete your SCU configuration, configure the output version of source class usage on the exit interface of your egress router:

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

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU with Layer 3 VPNs | 642
- SCU with Layer 3 VPNs Configuration | 661

Associating an Accounting Profile with SCU Classes

Once your source classes are defined, implemented on the inbound and outbound interfaces, and applied to the forwarding table, you are ready to associate the source class with an accounting profile. Configure the accounting profile at the `[edit accounting-options class-usage-profile]` hierarchy level. You can associate either an SCU source class or a DCU destination class with the accounting profile. You can also specify the filename for the data capture, a class usage profile name, and an interval (in minutes) indicating how often you want the SCU information to be saved to the file.

```
[edit]
accounting-options {
    file filename;
    class-usage-profile profile-name {
        file filename;
        interval minutes;
        source-classes {
            source-class-name;
        }
        destination-classes {
            destination-class-name;
        }
    }
}
```

NOTE: SCU accounting occurs on the outbound interface before output filter processing. If an SCU-marked packet is discarded in the router, the SCU counters can indicate more traffic than actually exists. You must use filter counters or traceoptions logs to ensure that all packets dropped by the SCU filter are recorded. If logged, you can subtract the discarded packets from the SCU counter tallies and calculate the true traffic profile.

Because DCU accounting occurs after the filtering process, DCU is unaffected by this disclaimer.

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639
- Roadmap for Configuring SCU with Layer 3 VPNs | 642
- SCU with Layer 3 VPNs Configuration | 661

Verifying Your SCU Accounting Profile

IN THIS SECTION

- Purpose | 649
- Action | 649

Purpose

To view the results of the SCU accounting profile you created.

Action

Navigate to the **/var/log** directory of your router. It should contain the designated class usage profile log. The layout of an SCU profile looks like this:

```
profile_name,epoch-timestamp,interface-name,source-class-name,packet-count,
byte-count
```

An example of the actual output from a profile looks like this:

```
scu_profile,980313078,ge-1/0/0.0,gold,82,6888
scu_profile,980313078,ge-1/0/0.0,silver,164,13776
scu_profile,980313078,ge-1/0/0.0,bronze,0,0
scu_profile,980313678,ge-1/0/0.0,gold,82,6888
scu_profile,980313678,ge-1/0/0.0,silver,246,20664
scu_profile,980313678,ge-1/0/0.0,bronze,0,0
```

To view the parameters of your SCU accounting profile, you can use the `show accounting-options class-usage-profile` *`scu-profile-name`* command.

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639

Associating an Accounting Profile with SCU Classes | 648

SCU Configuration

IN THIS SECTION

- Configuring SCU | 650
- Verifying Your Work | 654

Configuring SCU

Figure 39: SCU Topology Diagram

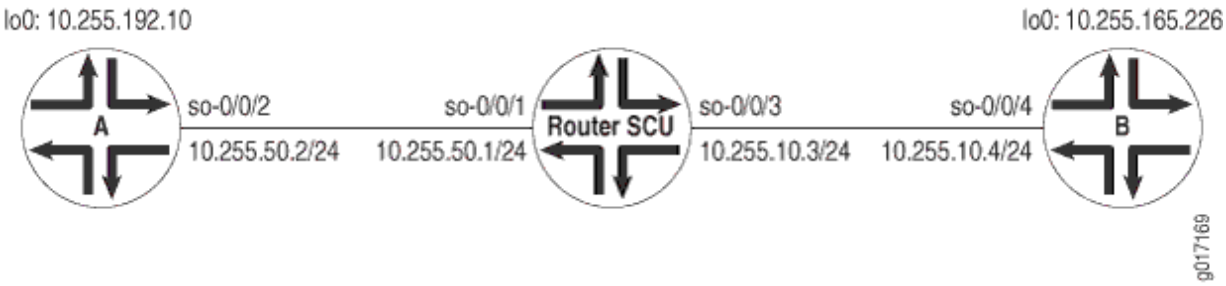


Figure 39 on page 650 shows a basic SCU configuration with three routers. Source routers A and B use loopback addresses as the prefixes to be monitored. Most of the configuration tasks and actual monitoring occurs on transit Router SCU.

Begin your configuration on Router A. The loopback address on Router A contains the origin of the prefix that is to be assigned to source class A on Router SCU. However, no SCU processing happens on this router. Therefore, configure Router A for basic OSPF routing and include your loopback interface and interface `so-0/0/2` in the OSPF process.

Router A:

```
[edit]
interfaces {
    so-0/0/2 {
        unit 0 {
```

```
            family inet {
                address 10.255.50.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.255.192.10/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface so-0/0/2.0;
            interface lo0.0;
        }
    }
}
```

Router SCU handles the bulk of the activity in this example. On Router SCU, enable source class usage on the inbound and outbound interfaces at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet accounting]` hierarchy level. Make sure you specify the expected traffic: input, output, or, in this case, both.

Next, configure a route filter policy statement that matches the prefixes of the loopback addresses from routers A and B. Include statements in the policy that classify packets from Router A in one group named `scu-class-a` and packets from Router B in a second class named `scu-class-b`. Notice the efficient use of a single policy containing multiple terms.

Last, apply the policy to the forwarding table.

Router SCU

```
[edit]
interfaces {
    so-0/0/1 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
```

```
                        input;
                        output;
                    }
                }
                address 10.255.50.1/24;
            }
        }
    }
    so-0/0/3 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
                        input;
                        output;
                    }
                }
                address 10.255.10.3/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.255.6.111/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface so-0/0/1.0;
            interface so-0/0/3.0;
        }
    }
}
routing-options {
    forwarding-table {
        export scu-policy;
    }
}
policy-options {
```

```
    policy-statement scu-policy {
        term 0 {
            from {
                route-filter 10.255.192.0/24 orlonger;
            }
            then source-class scu-class-a;
        }
        term 1 {
            from {
                route-filter 10.255.165.0/24 orlonger;
            }
            then source-class scu-class-b;
        }
    }
}
```

Complete the configuration tasks on Router B. Just as Router A provides a source prefix, Router B's loopback address matches the prefix assigned to `scu-class-b` on Router SCU. Again, no SCU processing happens on this router, so configure Router B for basic OSPF routing and include your loopback interface and interface `so-0/0/4` in the OSPF process.

Router B:

```
[edit]
interfaces {
    so-0/0/4 {
        unit 0 {
            family inet {
                address 10.255.10.4/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.255.165.226/32;
            }
        }
    }
}
protocols {
    ospf {
```

```
        area 0.0.0.0 {
            interface so-0/0/4.0;
            interface lo0.0;
        }
    }
}
```

Verifying Your Work

To verify that SCU is functioning properly, use the following commands:

- show interfaces *interface-name* statistics
- show interfaces *interface-name* (extensive | detail)
- show route (extensive | detail)
- show interfaces source-class *source-class-name interface-name*
- clear interface *interface-name* statistics

You should always verify SCU statistics at the outbound SCU interface on which you configured the output statement. You can perform the following three steps to check the functionality of SCU:

1. Clear all counters on your SCU-enabled router and verify that they are empty.
2. Send a ping from one edge router to another edge router to generate SCU traffic across the SCU-enabled router.
3. Verify that the counters are incrementing correctly on the outbound interface.

The following section shows the output of these commands as used with the configuration example.

```
user@scu> clear interfaces statistics all

user@scu>  show interfaces so-0/0/1.0 statistics
  Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                            Packets                Bytes
                      scu-class-a                  0                    0
                     scu-class-b                  0                    0
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.255.50/24, Local: 10.255.50.1
```

```
user@scu> show interfaces so-0/0/3.0 statistics
  Logical interface so-0/0/3.0 (Index 6) (SNMP ifIndex 113)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                            Packets               Bytes
                    scu-class-a                     0                   0
                    scu-class-b                     0                   0
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.255.10/24, Local: 10.255.10.3

user@scu> show interfaces source-class scu-class-a so-0/0/3.0
    Protocol inet
      Source class                            Packets               Bytes
                    scu-class-a                     0                   0

user@scu> show interfaces source-class scu-class-b so-0/0/1.0
    Protocol inet
      Source class                            Packets               Bytes
                    scu-class-b                     0                   0

user@routerB> ping 10.255.192.10 source 10.255.165.226 rapid 10000

user@routerA> ping 10.255.165.226 source 10.255.192.10 rapid 10000

user@scu> show interfaces source-class scu-class-a so-0/0/3.0
    Protocol inet
      Source class                            Packets               Bytes
                    scu-class-a                 20000             1680000

user@scu> show interfaces source-class scu-class-a so-0/0/1.0
    Protocol inet
      Source class                            Packets               Bytes
                    scu-class-b                 20000             1680000

user@scu> show interfaces so-0/0/3.0 statistics
  Logical interface so-0/0/3.0 (Index 6) (SNMP ifIndex 113)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                            Packets               Bytes
                    scu-class-a                 20000             1680000
                    scu-class-b                     0                   0
      Addresses, Flags: Is-Preferred Is-Primary
```

```
        Destination: 10.255.10/24, Local: 10.255.10.3

user@scu> show interfaces so-0/0/1.0 statistics
  Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                              Packets                Bytes
                      scu-class-a                     0                    0
                      scu-class-b                 20000              1680000
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.255.50/24, Local: 10.255.50.1
```

```
user@scu> show route extensive 10.255.192.0

inet.0: 26 destinations, 28 routes (25 active, 0 holddown, 1 hidden)
10.255.192.0/18 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.255.192.0/18 -> {so-0/0/1.0}
Source class: scu-class-a
        *OSPF   Preference: 150
                Next hop: via so-0/0/1.0, selected
                State: <Active Int Ext>
                Age: 2:49:31  Metric: 0  Tag: 0
                Task: OSPF
                Announcement bits (1): 0-KRT
                AS path: I
```

```
user@scu> show route extensive 10.255.165.0
inet.0: 26 destinations, 28 routes (25 active, 0 holddown, 1 hidden)
10.255.165.0/20 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.255.165.0/20 -> {so-0/0/3.0}
Source class: scu-class-b
        *OSPF   Preference: 150
                Next hop: via so-0/0/3.0, selected
                State: <Active Int Ext>
                Age: 2:49:31  Metric: 0  Tag: 0
```

```
                Task: OSPF
                Announcement bits (1): 0-KRT
                AS path: I
```

```
user@scu> show interfaces so-0/0/1 detail
Physical interface: so-0/0/1, Enabled, Physical link is Up
  Interface index: 12, SNMP ifIndex: 17, Generation: 11
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
  Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Link flags     : Keepalives
  Hold-times     : Up 0 ms, Down 0 ms
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive statistics:
    Input : 46 (last seen 00:00:01 ago)
    Output: 45 (last sent 00:00:00 ago)
  LCP state: Opened
  NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mpls:
  Not-configured
  CHAP state: Not-configured
  Last flapped   : 2002-04-19 11:49:22 PDT (03:10:09 ago)
  Statistics last cleared: 2002-04-19 14:52:04 PDT (00:07:27 ago)
  Traffic statistics:
   Input  bytes  :              1689276                   40 bps
   Output bytes  :              1689747                   48 bps
   Input  packets:                20197                    0 pps
   Output packets:                20200                    0 pps
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                20053                20053                    0
    1 expedited-fo                   0                    0                    0
    2 assured-forw                   0                    0                    0
    3 network-cont                 146                  146                    0
  SONET alarms   : None
  SONET defects  : None
  Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119) (Generation 3)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Flags: SCU-in, SCU-out
    Generation: 6 Route table: 0
```

```
        Source class                          Packets                Bytes
                      scu-class-a                   0                    0
                      scu-class-b               20000              1680000
      Filters: Input: icmp-so-0/0/1.0-i, Output: icmp-so-0/0/1.0-o
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.255.50/24, Local: 10.255.50.1, Broadcast: Unspecified,
        Generation: 8

user@scu>  show interfaces so-0/0/1 extensive
Physical interface: so-0/0/1, Enabled, Physical link is Up
  Interface index: 12, SNMP ifIndex: 17, Generation: 11
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
  Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Link flags     : Keepalives
  Hold-times     : Up 0 ms, Down 0 ms
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive statistics:
    Input : 51 (last seen 00:00:04 ago)
    Output: 50 (last sent 00:00:05 ago)
  LCP state: Opened
  NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mpls:
  Not-configured
  CHAP state: Not-configured
  Last flapped   : 2002-04-19 11:49:22 PDT (03:11:05 ago)
  Statistics last cleared: 2002-04-19 14:52:04 PDT (00:08:23 ago)
  Traffic statistics:
   Input  bytes  :              1689884                 264 bps
   Output bytes  :              1690388                 280 bps
   Input  packets:                20215                   0 pps
   Output packets:                20217                   0 pps
  Input errors:
    Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Giants: 0,
    Bucket drops: 0, Policed discards: 0, L3 incompletes: 0,
    L2 channel errors: 0, L2 mismatch timeouts: 0, HS link CRC errors: 0,
    HS link FIFO overflows: 0
  Output errors:
    Carrier transitions: 0, Errors: 0, Drops: 0, Aged packets: 0,
    HS link FIFO underflows: 0
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0 best-effort                20053                20053                    0
    1 expedited-fo                   0                    0                    0
```

```
    2 assured-forw                  0                   0                   0
    3 network-cont                164                 164                   0
  SONET alarms   : None
  SONET defects  : None
  SONET PHY:            Seconds        Count  State
    PLL Lock                  0            0  OK
    PHY Light                 0            0  OK
  SONET section:
    BIP-B1                    0            0
    SEF                       0            0  OK
    LOS                       0            0  OK
    LOF                       0            0  OK
    ES-S                      0
    SES-S                     0
    SEFS-S                    0
  SONET line:
    BIP-B2                    0            0
    REI-L                     0            0
    RDI-L                     0            0  OK
    AIS-L                     0            0  OK
    BERR-SF                   0            0  OK
    BERR-SD                   0            0  OK
    ES-L                      0
    SES-L                     0
    UAS-L                     0
    ES-LFE                    0
    SES-LFE                   0
    UAS-LFE                   0
  SONET path:
    BIP-B3                    0            0
    REI-P                     0            0
    LOP-P                     0            0  OK
    AIS-P                     0            0  OK
    RDI-P                     0            0  OK
    UNEQ-P                    0            0  OK
    PLM-P                     0            0  OK
    ES-P                      0
    SES-P                     0
    UAS-P                     0
    ES-PFE                    0
    SES-PFE                   0
    UAS-PFE                   0
  Received SONET overhead:
```

```
  F1      : 0x00, J0      : 0x00, K1      : 0x00, K2      : 0x00
  S1      : 0x00, C2      : 0xcf, C2(cmp) : 0xcf, F2      : 0x00
  Z3      : 0x00, Z4      : 0x00, S1(cmp) : 0x00, V5      : 0x00
  V5(cmp) : 0x00
Transmitted SONET overhead:
  F1      : 0x00, J0      : 0x01, K1      : 0x00, K2      : 0x00
  S1      : 0x00, C2      : 0xcf, F2      : 0x00, Z3      : 0x00
  Z4      : 0x00, V5      : 0x00
Received path trace: e so-0/0/1
  65 20 73 6f 2d 30 2f 30 2f 31 00 00 00 00 00 00   e so-0/0/1......
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 0a   ................
Transmitted path trace: scu so-0/0/1
  67 68 62 20 73 6f 2d 30 2f 30 2f 31 00 00 00 00   scu so-0/0/1....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
HDLC configuration:
  Policing bucket: Disabled
  Shaping bucket : Disabled
  Giant threshold: 4484, Runt threshold: 3
Packet Forwarding Engine configuration:
  Destination slot: 0, PLP byte: 1 (0x00)
  CoS transmit queue          Bandwidth          Buffer     Priority   Limit
                             %          bps   %        bytes
  0 best-effort              0            0   0            0      low    none
  1 expedited-forwarding     0            0   0            0      low    none
  2 assured-forwarding       0            0   0            0      low    none
  3 network-control          0            0   0            0      low    none
Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119) (Generation 3)
  Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
  Protocol inet, MTU: 4470
    Flags: SCU-in, SCU-out
  Generation: 6 Route table: 0
     Source class                            Packets                Bytes
                  scu-class-a                     0                    0
                  scu-class-b                 20000              1680000
    Filters: Input: icmp-so-0/0/1.0-i, Output: icmp-so-0/0/1.0-o
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.255.50/24, Local: 10.255.50.1, Broadcast: Unspecified,
      Generation: 8
```

RELATED DOCUMENTATION

Source Class Usage Overview | 637
System Requirements for SCU | 639
Roadmap for Configuring SCU | 641

SCU with Layer 3 VPNs Configuration

IN THIS SECTION

Configuring SCU in a Layer 3 VPN | 661
Verifying Your Work | 670

Configuring SCU in a Layer 3 VPN

Figure 40: SCU in a Layer 3 VPN Topology Diagram

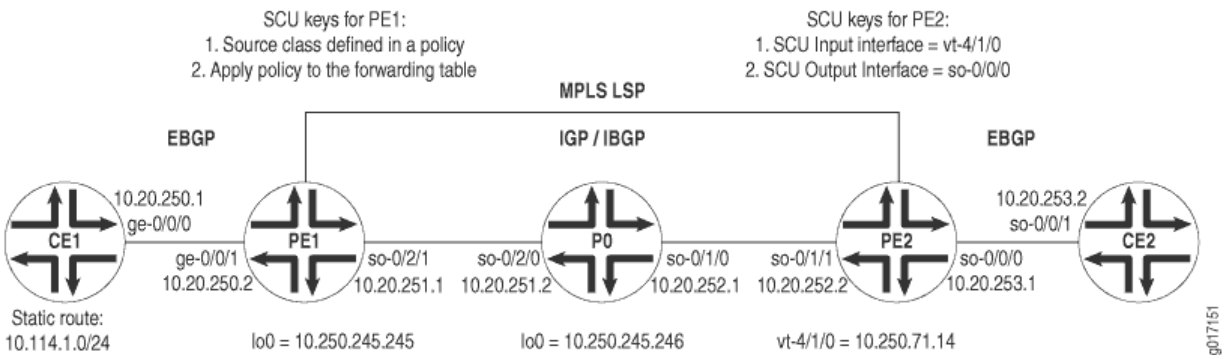


Figure 40 on page 661 displays a Layer 3 VPN topology. CE1 and CE2 are customer edge (CE) routers connected by a VPN through provider routers PE1, P0, and PE2. EBGP is established between routers CE1 and PE1, IBGP connects routers PE1 and PE2 over an IS-IS/MPLS/LDP core, and a second EBGP connection flows between routers PE2 and CE2.

On Router CE1, begin your VPN by setting up an EBGP connection to PE1. Install a static route of `10.114.1.0/24` and advertise this route to your EBGP neighbor.

Router CE1

```
[edit]
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                address 10.20.250.1/30;
            }
        }
    }
}
routing-options {
    static {
        route 10.114.1.0/24 reject;
    }
    autonomous-system 100;
}
protocols {
    bgp {
        group to-pe1 {
            local-address 10.20.250.1;
            export inject-direct;
            peer-as 300;
            neighbor 10.20.250.2;
        }
    }
}
policy-options {
    policy-statement inject-direct {
        term 1 {
            from {
                protocol static;
                route-filter 10.114.1.0/24 exact;
            }
            then accept;
        }
        term 2 {
            from protocol direct;
            then accept;
        }
```

```
    }
}
```

On PE1, complete the EBGP connection to CE1 through a VRF routing instance. Set an export policy for your VRF instance that puts BGP traffic into a community, and an import policy that accepts like community traffic from your VPN neighbor. Lastly, configure an IBGP relationship to Router PE2 that runs over an IS-IS, MPLS, and LDP core.

Router PE1

```
[edit]
interfaces {
    ge-0/0/1 {
        unit 0 {
            family inet {
                address 10.20.250.2/30;
            }
        }
    }
    so-0/2/1 {
        unit 0 {
            family inet {
                address 10.20.251.1/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.245.245/32;
            }
            family iso;
            family mpls;
        }
    }
}
routing-options {
    autonomous-system 300;
}
protocols {
```

```
    mpls {
        interface so-0/2/1;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.250.245.245;
            family inet-vpn {
                unicast;
            }
            neighbor 10.250.71.14;
        }
    }
    isis {
        interface so-0/2/1;
    }
    ldp {
        interface so-0/2/1;
    }
}
policy-options {
    policy-statement red-import {
        from {
            protocol bgp;
            community red-com;
        }
        then accept;
    }
    policy-statement red-export {
        from protocol bgp;
        then {
            community add red-com;
            accept;
        }
    }
    community red-com members target:20:20;
}
routing-instances {
    red {
        instance-type vrf;
        interface ge-0/0/1.0;
        route-distinguisher 10.250.245.245:100;
        vrf-import red-import;
```

```
        vrf-export red-export;
        protocols {
            bgp {
                group to-ce1 {
                    local-address 10.20.250.2;
                    peer-as 100;
                    neighbor 10.20.250.1;
                }
            }
        }
    }
}
```

On P0, connect the IBGP neighbors located at PE1 and PE2. Remember to include VPN-related protocols (MPLS, LDP, and IGP) on all interfaces.

Router P0

```
[edit]
interfaces {
    so-0/1/0 {
        unit 0 {
            family inet {
                address 10.20.252.1/30;
            }
            family iso;
            family mpls;
        }
    }
    so-0/2/0 {
        unit 0 {
            family inet {
                address 10.20.251.2/30;
            }
            family iso;
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.245.246/32;
```

```
            }
            family iso;
            family mpls;
        }
    }
}
routing-options {
    autonomous-system 300;
}
protocols {
    mpls {
        interface so-0/1/0;
        interface so-0/2/0;
    }
    isis {
        interface all;
    }
    ldp {
        interface all;
    }
}
```

On PE2, complete the IBGP relationship to Router PE1. Establish an EBGP connection to CE2 through a VRF routing instance. Set an export policy for the VRF instance that places BGP traffic into a community, and an import policy that accepts like community traffic from the VPN neighbor. Next, establish a policy that adds the static route from CE1 to a source class called `GOLD1`. Also, export this SCU policy into the forwarding table. Finally, set your `vt` interface as the SCU input interface and establish the CE-facing interface `so-0/0/0` as the SCU output interface.

Router PE2

```
[edit]
interfaces {
    so-0/1/1 {
        unit 0 {
            family inet {
                address 10.20.252.2/30;
            }
            family iso;
            family mpls;
        }
    }
```

```
    so-0/0/0 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
                        output;
                    }
                }
                address 10.20.253.1/30;
            }
        }
    }
    vt-4/1/0 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
                        input;
                    }
                }
                address 10.250.71.14/32;
            }
            family iso;
            family mpls;
        }
    }
}
routing-options {
    autonomous-system 300;
    forwarding-table {
        export inject-customer2-dest-class;
    }
}
protocols {
    mpls {
        interface so-0/1/1;
        interface vt-4/1/0;
    }
    bgp {
        group ibgp {
            type internal;
            local-address 10.250.71.14;
            family inet-vpn {
```

```
                unicast;
            }
            neighbor 10.250.245.245;
        }
    }
    isis {
        interface so-0/1/1;
    }
    ldp {
        interface so-0/1/1;
    }
}
routing-instances {
    red {
        instance-type vrf;
        interface so-0/0/0.0;
        interface vt-4/1/0.0;
        route-distinguisher 10.250.71.14:100;
        vrf-import red-import;
        vrf-export red-export;
        protocols {
            bgp {
                group to-ce2 {
                    local-address 10.20.253.1;
                    peer-as 400;
                    neighbor 10.20.253.2;
                }
            }
        }
    }
}
policy-options {
    policy-statement red-import {
        from {
            protocol bgp;
            community red-com;
        }
        then accept;
    }
    policy-statement red-export {
        from protocol bgp;
        then {
            community add red-com;
```

```
            accept;
        }
    }
    policy-statement inject-customer2-dest-class {
        term term-gold1-traffic {
            from {
                route-filter 10.114.1.0/24 exact;
            }
            then source-class GOLD1;
        }
    }
    community red-com members target:20:20;
}
```

On Router CE2, complete the VPN path by finishing the EBGP connection to PE2.

Router CE2

```
[edit]
interfaces {
    so-0/0/1 {
        unit 0 {
            family inet {
                address 10.20.253.2/30;
            }
        }
    }
}
routing-options {
    autonomous-system 400;
}
protocols {
    bgp {
        group to-pe2 {
            local-address 10.20.253.2;
            export inject-direct;
            peer-as 300;
            neighbor 10.20.253.1;
        }
    }
}
policy-options {
```

```
    policy-statement inject-direct {
        from {
            protocol direct;
        }
        then accept;
    }
}
```

Verifying Your Work

To verify that SCU is functioning properly in the Layer 3 VPN, use the following commands:

- show interfaces *interface-name* statistics
- show interfaces source-class *source-class-name interface-name*
- show interfaces *interface-name* (extensive | detail)
- show route (extensive | detail)
- clear interface *interface-name* statistics

You should always verify SCU statistics at the outbound SCU interface on which you configured the output statement. To check SCU functionality, follow these steps:

1. Clear all counters on your SCU-enabled router and verify they are empty.
2. Send a ping from the ingress CE router to the second CE router to generate SCU traffic across the SCU-enabled VPN route.
3. Verify that the counters are incrementing correctly on the outbound interface.

The following section shows the output of these commands used with the configuration example.

```
user@pe2> clear interfaces statistics all

    user@pe2> show interfaces so-0/0/0.0 statistics
      Logical interface so-0/0/0.0 (Index 6) (SNMP ifIndex 113)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                              Packets                Bytes
 GOLD1                          0                    0
      Addresses, Flags: Is-Preferred Is-Primary
```

```
    user@pe2> show interfaces source-class GOLD1 so-0/0/0.0
    Protocol inet
      Source class                             Packets                Bytes
GOLD1                          0                   0

user@ce1> ping 10.20.253.2 source 10.114.1.1 rapid count 10000

user@scu>  show interfaces source-class GOLD1 so-0/0/0.0
    Protocol inet
      Source class                             Packets                Bytes
GOLD1                       20000              1680000

user@scu>  show interfaces so-0/0/0.0 statistics
  Logical interface so-0/0/0.0 (Index 6) (SNMP ifIndex 113)
    Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
    Protocol inet, MTU: 4470
      Source class                             Packets                Bytes
GOLD1                       20000              1680000
      Addresses, Flags: Is-Preferred Is-Primary
         Destination: 10.20.253/24, Local: 10.20.253.1
```

RELATED DOCUMENTATION

- Source Class Usage Overview | 637
- System Requirements for SCU | 639

Example: Grouping Source and Destination Prefixes into a Forwarding Class

IN THIS SECTION

- Requirements | 672
- Overview | 672
- Configuration | 675
- Verification | 682

This example shows how to group source and destination prefixes into a forwarding class.

Requirements

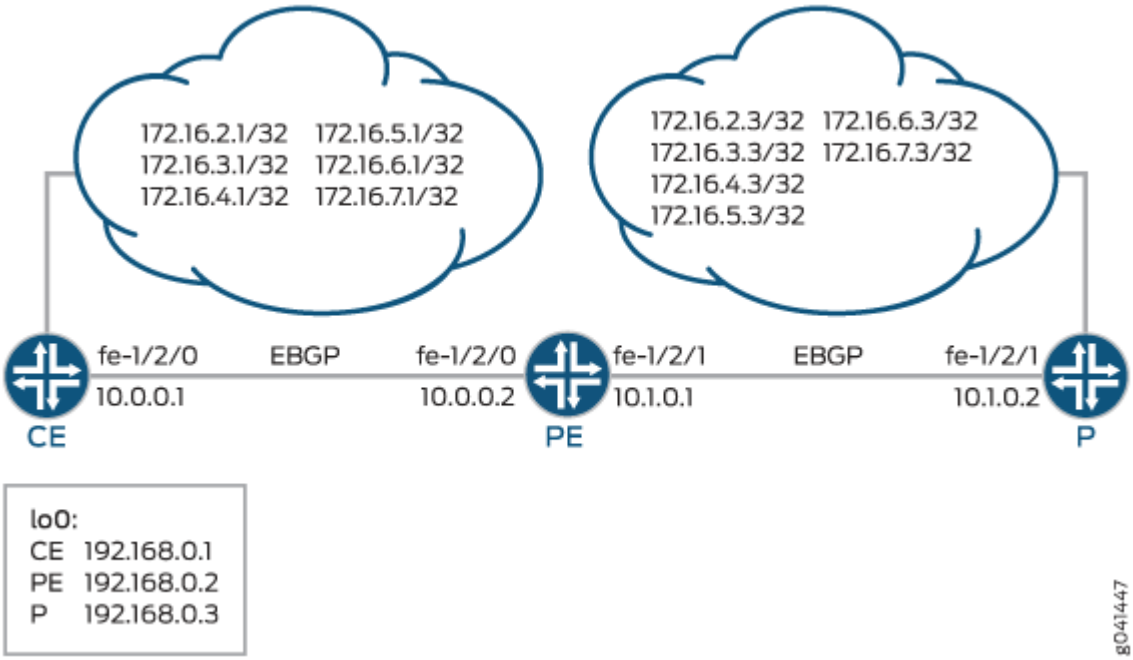
No special configuration beyond device initialization is required before configuring this example.

Overview

This example uses three routing devices: a customer edge (CE) device, a provider edge (PE) device, and a provider core (P) device.

Figure 41 on page 672 shows the sample network.

Figure 41: SCU and DCU Sample Network



Source class usage (SCU) counts packets sent to the customer edge by performing lookup on the IP source address and the IP destination address. SCU makes it possible to track traffic originating from specific prefixes on the provider core and destined for specific prefixes on the customer edge.

DCU counts packets from customers by performing a lookup of the IP destination address. DCU makes it possible to track traffic originating from the customer edge and destined for specific prefixes on the provider core router.

On Device PE's fe-1/2/1 interface, facing the provider core (represented by Device P), SCU input is configured with the `source-class-usage input` statement to track traffic originating at Device P and destined

to Device CE. On this same interface, the `destination-class-usage input` statement is configured to track traffic originating at Device CE destined to the provider core.

```
user@PE# show interfaces fe-1/2/1 unit 0 family inet
accounting {
    source-class-usage {
        input;  # tracks traffic destined to customer edge
    }
    destination-class-usage; # tracks traffic destined to provider core
}
address 10.1.0.1/30;
```

Unlike destination class usage (DCU), which only requires implementation on a single interface, accounting for SCU must be enabled on two interfaces: the inbound and outbound interfaces traversed by the source class. You must define explicitly the two interfaces on which SCU monitored traffic is expected to arrive and depart. This is because SCU performs two lookups in the routing table: a source address (SA) and a destination address (DA) lookup. In contrast, DCU only has a single destination address lookup.

On Device PE's fe-1/2/0 interface, facing Device CE, SCU output is configured with the `source-class-usage output` statement.

```
user@PE# show interfaces fe-1/2/0 unit 0 family inet
accounting {
    source-class-usage {
        output;
    }
}
address 10.0.0.2/30;
```

To account for traffic destined to the customer, the policy called scu_class uses route filters to place traffic into the gold1, gold2, and gold3 classes.

```
user@PE# show policy-options
policy-statement scu_class {
    term gold1 {
        from {
            route-filter 172.16.2.0/24 orlonger;
        }
        then source-class gold1;
    }
```

```
    term gold2 {
        from {
            route-filter 172.16.3.0/24 orlonger;
        }
        then source-class gold2;
    }
    term gold3 {
        from {
            route-filter 172.16.4.0/24 orlonger;
        }
        then source-class gold3;
    }
}
```

To account for traffic destined to the provider, the policy called dcu_class uses route filters to place traffic into the silver1, silver2, and silver3 classes.

```
user@PE# show policy-options
policy-statement dcu_class {
    term silver1 {
        from {
            route-filter 172.16.5.0/24 orlonger;
        }
        then destination-class silver1;
    }
    term silver2 {
        from {
            route-filter 172.16.6.0/24 orlonger;
        }
        then destination-class silver2;
    }
    term silver3 {
        from {
            route-filter 172.16.7.0/24 orlonger;
        }
        then destination-class silver3;
    }
}
```

The policies are then applied to the forwarding table.

```
forwarding-table {
    export [ dcu_class scu_class ];
}
```

The example uses static routes to provide connectivity and loopback interface addresses for testing the operation.

"CLI Quick Configuration" on page 675 shows the configuration for all of the devices in Figure 41 on page 672.

The section "No Link Title" on page 678 describes the steps on Device PE.

Configuration

IN THIS SECTION

- Procedure | **675**

Procedure

CLI Quick Configuration

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

Device CE

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set protocols bgp group ext type external
```

```
set protocols bgp group ext export send-direct
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.1.0.0/30 next-hop 10.0.0.2
set routing-options autonomous-system 100
```

Device PE

```
set interfaces fe-1/2/0 unit 0 family inet accounting source-class-usage output
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 family inet accounting source-class-usage input
set interfaces fe-1/2/1 unit 0 family inet accounting destination-class-usage
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
set policy-options policy-statement dcu_class term silver1 from route-filter 172.16.5.0/24
orlonger
set policy-options policy-statement dcu_class term silver1 then destination-class silver1
set policy-options policy-statement dcu_class term silver2 from route-filter 172.16.6.0/24
orlonger
set policy-options policy-statement dcu_class term silver2 then destination-class silver2
set policy-options policy-statement dcu_class term silver3 from route-filter 172.16.7.0/24
orlonger
set policy-options policy-statement dcu_class term silver3 then destination-class silver3
set policy-options policy-statement scu_class term gold1 from route-filter 172.16.2.0/24 orlonger
set policy-options policy-statement scu_class term gold1 then source-class gold1
set policy-options policy-statement scu_class term gold2 from route-filter 172.16.3.0/24 orlonger
set policy-options policy-statement scu_class term gold2 then source-class gold2
set policy-options policy-statement scu_class term gold3 from route-filter 172.16.4.0/24 orlonger
set policy-options policy-statement scu_class term gold3 then source-class gold3
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options autonomous-system 200
```

```
set routing-options forwarding-table export dcu_class
set routing-options forwarding-table export scu_class
```

Device P

```
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.1.0.1
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.0.0.0/30 next-hop 10.1.0.1
set routing-options static route 172.16.2.0/24 discard
set routing-options static route 172.16.3.0/24 discard
set routing-options static route 172.16.4.0/24 discard
set routing-options static route 172.16.5.0/24 discard
set routing-options static route 172.16.6.0/24 discard
set routing-options static route 172.16.7.0/24 discard
set routing-options autonomous-system 300
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To group source and destination prefixes in a forwarding class:

1. Create the router interfaces.

```
[edit interfaces]
user@PE# set fe-1/2/0 unit 0 family inet accounting source-class-usage output
user@PE# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@PE# set fe-1/2/1 unit 0 family inet accounting source-class-usage input
user@PE# set fe-1/2/1 unit 0 family inet accounting destination-class-usage
user@PE# set fe-1/2/1 unit 0 family inet address 10.1.0.1/30
user@PE# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure BGP.

```
[edit protocols bgp group ext]
user@PE# set type external
user@PE# set export send-direct
user@PE# set neighbor 10.0.0.1 peer-as 100
user@PE# set neighbor 10.1.0.2 peer-as 300
```

3. Configure the DCU policy.

```
[edit policy-options policy-statement dcu_class]
user@PE# set term silver1 from route-filter 172.16.5.0/24 orlonger
user@PE# set term silver1 then destination-class silver1
user@PE# set term silver2 from route-filter 172.16.6.0/24 orlonger
user@PE# set term silver2 then destination-class silver2
user@PE# set term silver3 from route-filter 172.16.7.0/24 orlonger
user@PE# set term silver3 then destination-class silver3
```

4. Configure the SCU policy.

```
[edit policy-options policy-statement scu_class]
user@PE# set term gold1 from route-filter 172.16.2.0/24 orlonger
user@PE# set term gold1 then source-class gold1
user@PE# set term gold2 from route-filter 172.16.3.0/24 orlonger
user@PE# set term gold2 then source-class gold2
user@PE# set term gold3 from route-filter 172.16.4.0/24 orlonger
user@PE# set term gold3 then source-class gold3
```

5. Apply the policies to the forwarding table.

```
[edit routing-options forwarding-table]
user@PE# set export dcu_class
user@PE# set export scu_class
```

> **NOTE**: You can refer to the same routing policy one or more times in the same or different export statement.

6. (Optional) Configure a routing policy that advertises direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@PE# set from protocol direct
user@PE# set then accept
```

7. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@PE# set autonomous-system 200
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage {
                    output;
                }
            }
            address 10.0.0.2/30;
        }
```

```
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage {
                    input;
                }
                destination-class-usage;
            }
            address 10.1.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@PE# show protocols
bgp {
    group ext {
        type external;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
        neighbor 10.1.0.2 {
            peer-as 300;
        }
    }
}
```

```
user@PE# show policy-options
policy-statement dcu_class {
    term silver1 {
```

```
        from {
            route-filter 172.16.5.0/24 orlonger;
        }
        then destination-class silver1;
    }
    term silver2 {
        from {
            route-filter 172.16.6.0/24 orlonger;
        }
        then destination-class silver2;
    }
    term silver3 {
        from {
            route-filter 172.16.7.0/24 orlonger;
        }
        then destination-class silver3;
    }
}
policy-statement scu_class {
    term gold1 {
        from {
            route-filter 172.16.2.0/24 orlonger;
        }
        then source-class gold1;
    }
    term gold2 {
        from {
            route-filter 172.16.3.0/24 orlonger;
        }
        then source-class gold2;
    }
    term gold3 {
        from {
            route-filter 172.16.4.0/24 orlonger;
        }
        then source-class gold3;
    }
}
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
```

```
    }
}
```

```
user@PE# show routing-options
autonomous-system 200;
forwarding-table {
    export [ dcu_class scu_class ];
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Making Sure That the DCU Policy Is Working | **682**
- Making Sure That the SCU Policy Is Working | **683**

Confirm that the configuration is working properly.

Making Sure That the DCU Policy Is Working

Purpose

Verify that traffic sent from the provider core into the customer network is causing the DCU policy counters to increment.

Action

1. From Device P, ping an address in the customer network.

```
user@P> ping rapid count 10000000 172.16.0.1

PING 172.16.0.1 (6.0.0.1): 56 data bytes
!!!
```

2. On Device PE, check the interface statistics on the interface facing the provider core.

```
user@PE> show interfaces statistics fe-1/2/1.0

  Logical interface fe-1/2/1.0 (Index 108) (SNMP ifIndex 546)
    Flags: SNMP-Traps 0x4000 Encapsulation: ENET2
    Input packets : 251956
    Output packets: 251961
    Protocol inet, MTU: 1500
      Flags: Sendbcast-pkt-to-re, DCU, SCU-in
                                              Packets                 Bytes
      Destination class              (packet-per-second)    (bits-per-second)

                           silver1                  7460                626640
                                   (                   0) (                  0)
                           silver2                 22440               2401416
                                   (                 256) (             171963)
                           silver3                  9004                756336
                                   (                   0) (                  0)
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.1.0.0/30, Local: 10.1.0.1, Broadcast: 10.1.0.3
```

Meaning

Packet and bit rates are displayed with packet and byte counters.

Alternatively, you can use the `show interfaces destination-class all` command to display the same information.

Making Sure That the SCU Policy Is Working

Purpose

Verify that traffic sent from the customer network into the provider core is causing the SCU policy counters to increment.

Action

1. From Device CE, ping an address in the customer network.

```
user@CE> ping rapid count 10000000 172.16.0.1

PING 172.16.0.1 (6.0.0.1): 56 data bytes
!!!
```

```
        Addresses, Flags: Is-Preferred Is-Primary
          Destination: 10.0.0.0/30, Local: 10.0.0.2, Broadcast: 10.0.0.3
```

Meaning

Packet and bit rates are displayed with packet and byte counters.

Alternatively, you can use the `show interfaces source-class all` command to display the same information.

RELATED DOCUMENTATION

Route Filter Match Conditions | 72

Understanding Source Class Usage and Destination Class Usage Options

CHAPTER 9

Avoiding Traffic Routing Threats with Conditional Routing Policies

IN THIS CHAPTER

- Conditional Advertisement and Import Policy (Routing Table) with certain match conditions | 686
- Conditional Advertisement Enabling Conditional Installation of Prefixes Use Cases | 689
- Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table | 691

Conditional Advertisement and Import Policy (Routing Table) with certain match conditions

BGP accepts all non-looped routes learned from neighbors and imports them into the RIB-In table. If these routes are accepted by the BGP import policy, they are then imported into the inet.0 routing table. In cases where only certain routes are required to be imported, provisions can be made such that the peer routing device exports routes based on a condition or a set of conditions.

The condition for exporting a route can be based on:

- The peer the route was learned from
- The interface the route was learned on
- Some other required attribute

For example:

```
[edit]
policy-options {
    condition condition-name {
        if-route-exists address table table-name;
```

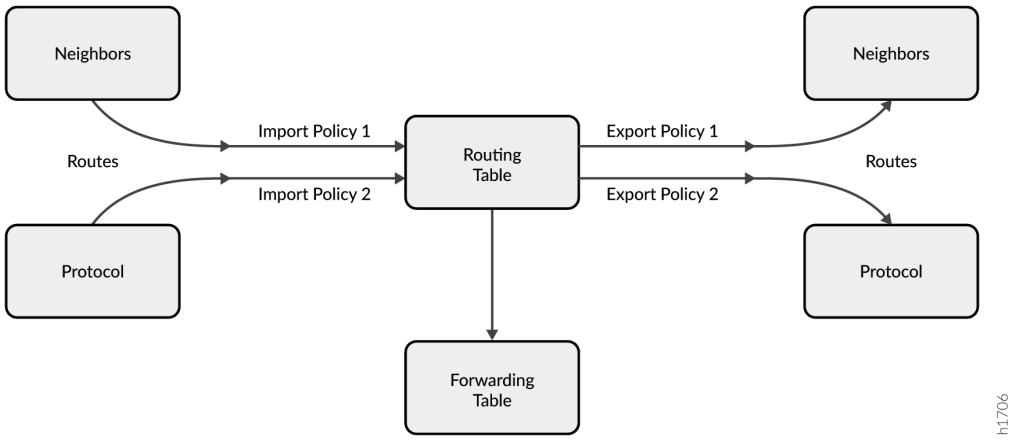
```
    }
}
```

This is known as conditional installation of prefixes and is described in *Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table*.

Conditions in routing policies can be configured irrespective of whether they are a part of the export or import policies or both. The export policy supports these conditions inherited from the routing policy based on the existence of another route in the routing policy. However, the import policy doesn't support these conditions, and the conditions are not executed even if they are present.

Figure 42 on page 687 illustrates where BGP import and export policies are applied. An import policy is applied to inbound routes that are visible in the output of the `show route receive-protocol bgp` *`neighbor-address`* command. An export policy is applied to outbound routes that are visible in the output of the `show route advertising-protocol bgp` *`neighbor-address`* command.

Figure 42: BGP Import and Export Policies



To enable conditional installation of prefixes, an export policy must be configured on the device where the prefix export has to take place. The export policy evaluates each route to verify that it satisfies all the match conditions under the `from` statement. It also searches for the existence of the route defined under the `condition` statement (also configured under the `from` statement).

If the route does not match the entire set of required conditions defined in the policy, or if the route defined under the `condition` statement does not exist in the routing table, the route is not exported to its BGP peers. Thus, a conditional export policy matches the routes for the desired route or prefix you want installed in the peers' routing table.

To configure the conditional installation of prefixes with the help of an export policy:

1. Create a `condition` statement to check prefixes.

```
[edit]
policy-options {
    condition condition-name {
        if-route-exists address table table-name;
    }
}
```

2. Create an export policy with the newly created condition using the `condition` statement.

```
[edit]
policy-options {
    policy-statement policy-name {
        term 1 {
            from {
                protocols bgp;
                condition condition-name;
            }
            then {
                accept;
            }
        }
    }
}
```

3. Apply the export policy to the device that requires only selected prefixes to be exported from the routing table.

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

RELATED DOCUMENTATION

Conditional Advertisement Enabling Conditional Installation of Prefixes Use Cases

Conditional Advertisement Enabling Conditional Installation of Prefixes Use Cases

Networks are usually subdivided into smaller, more-manageable units called autonomous systems (ASs). When BGP is used by routers to form peer relationships in the same AS, it is referred to as internal BGP (IBGP). When BGP is used by routers to form peer relationships in different ASs, it is referred to as external BGP (EBGP).

After performing route sanity checks, a BGP router accepts the routes received from its peers and installs them into the routing table. By default, all routers in IBGP and EBGP sessions follow the standard BGP advertisement rules. While a router in an IBGP session advertises only the routes learned from its direct peers, a router in an EBGP session advertises all routes learned from its direct and indirect peers (peers of peers). Hence, in a typical network configured with EBGP, a router adds all routes received from an EBGP peer into its routing table and advertises nearly all routes to all EBGP peers.

A service provider exchanging BGP routes with both customers and peers on the Internet is at risk of malicious and unintended threats that can compromise the proper routing of traffic, as well as the operation of the routers.

This has several disadvantages:

- **Non-aggregated route advertisements**—A customer could erroneously advertise all its prefixes to the ISP rather than an aggregate of its address space. Given the size of the Internet routing table, this must be carefully controlled. An edge router might also need only a default route out toward the Internet and instead be receiving the entire BGP routing table from its upstream peer.
- **BGP route manipulation**—If a malicious administrator alters the contents of the BGP routing table, it could prevent traffic from reaching its intended destination.
- **BGP route hijacking**—A rogue administrator of a BGP peer could maliciously announce a network's prefixes in an attempt to reroute the traffic intended for the victim network to the administrator's network to either gain access to the contents of traffic or to block the victim's online services.
- **BGP denial of service (DoS)**—If a malicious administrator sends unexpected or undesirable BGP traffic to a router in an attempt to use all of the router's available BGP resources, it might result in impairing the router's ability to process valid BGP route information.

Conditional installation of prefixes can be used to address all the problems previously mentioned. If a customer requires access to remote networks, it is possible to install a specific route in the routing table

of the router that is connected with the remote network. This does not happen in a typical EBGP network and hence, conditional installation of prefixes becomes essential.

ASs are not only bound by physical relationships but by business or other organizational relationships. An AS can provide services to another organization, or act as a transit AS between two other ASs. These transit ASs are bound by contractual agreements between the parties that include parameters on how to connect to each other and most importantly, the type and quantity of traffic they carry for each other. Therefore, for both legal and financial reasons, service providers must implement policies that control how BGP routes are exchanged with neighbors, which routes are accepted from those neighbors, and how those routes affect the traffic between the ASs.

There are many different options available to filter routes received from a BGP peer to both enforce inter-AS policies and mitigate the risks of receiving potentially harmful routes. Conventional route filtering examines the attributes of a route and accepts or rejects the route based on such attributes. A policy or filter can examine the contents of the AS-Path, the next-hop value, a community value, a list of prefixes, the address family of the route, and so on.

In some cases, the standard “acceptance condition” of matching a particular attribute value is not enough. The service provider might need to use another condition outside of the route itself, for example, another route in the routing table. As an example, it might be desirable to install a default route received from an upstream peer, only if it can be verified that this peer has reachability to other networks further upstream. This conditional route installation avoids installing a default route that is used to send traffic toward this peer, when the peer might have lost its routes upstream, leading to black-holed traffic. To achieve this, the router can be configured to search for the presence of a particular route in the routing table, and based on this knowledge accept or reject another prefix.

Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table explains how the conditional installation of prefixes can be configured and verified.

RELATED DOCUMENTATION

> *Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table*

Example: Configuring a Routing Policy for Conditional Advertisement Enabling Conditional Installation of Prefixes in a Routing Table

IN THIS SECTION

- Requirements | 691
- Overview | 691
- Configuration | 695
- Verification | 705

This example shows how to configure conditional installation of prefixes in a routing table using BGP export policy.

Requirements

This example uses the following hardware and software components:

- M Series Multiservice Edge Routers, MX Series 5G Universal Routing Platforms, or T Series Core Routers
- Junos OS Release 9.0 or later

Overview

IN THIS SECTION

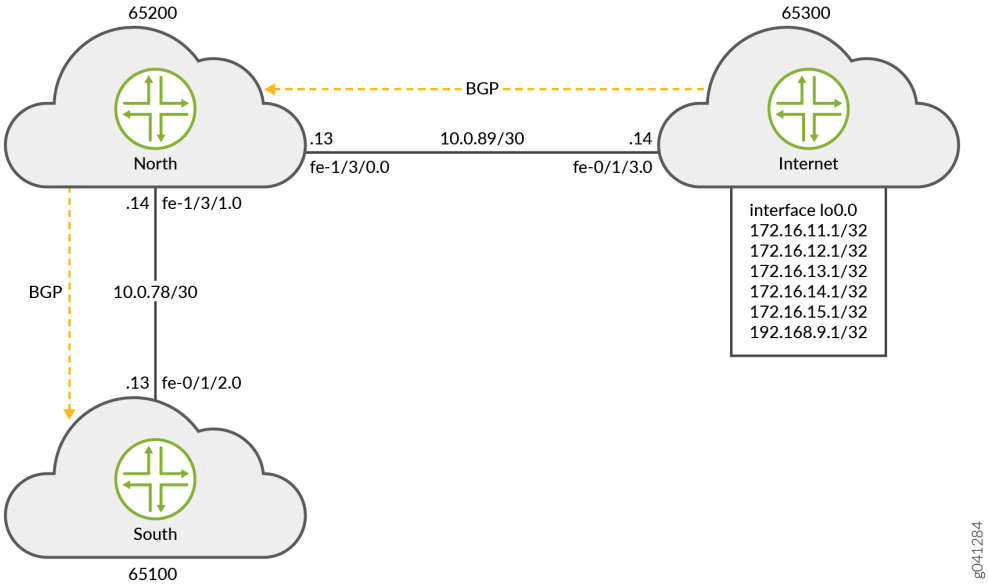
- Topology | 695

In this example, three routers in three different autonomous systems (ASs) are connected and configured with the BGP protocol. The router labeled Internet, which is the upstream router, has five addresses configured on its lo0.0 loopback interface (172.16.11.1/32, 172.16.12.1/32, 172.16.13.1/32, 172.16.14.1/32, and 172.16.15.1/32), and an extra loopback address (192.168.9.1/32) is configured as the router ID. These six addresses are exported into BGP to emulate the contents of a BGP routing table of a router connected to the Internet, and advertised to North.

The North and South routers use the 10.0.89.12/30 and 10.0.78.12/30 networks, respectively, and use 192.168.7.1 and 192.168.8.1 for their respective loopback addresses.

Figure 43 on page 692 shows the topology used in this example.

Figure 43: Conditional Installation of Prefixes



Router North exports the 172.16.0.0/16 BGP routes it learns from Router Internet to Router South. These routes might represent the routes owned by the Internet router's domain. In addition, when the specific 172.16.11.1/32 route is present, Router North also advertises a default route. The 172.16.11.1 route might represent the Internet router's link to a tier 1 transit peering provider that provides full internet connectivity.

Router South receives all six routes, but should only install the default route and one other specific route (172.16.11.1/32) in its routing table.

To summarize, the example meets the following requirements:

- On North, send all 172.16/16 prefixes. In addition, also send 0/0 to South only if a particular route is present in the inet.0 routing table (in this example 172.16.11.1/32).
- On South, accept and install the default route and the 172.16.11.1/32 route in the routing and forwarding tables. Drop all other routes. Consider that South might be a lower-end device that cannot accept a full Internet routing table. As a result the operator only wants South to have the default route and one other specific prefix.

The first requirement is met with an export policy on North:

```
user@North# show policy-options
policy-statement conditional-export-bgp {
    term prefix_11 {
        from {
            protocol bgp;
            route-filter 172.16.0.0/16 orlonger;
        }
        then accept;
    }
    term conditional-default {
        from {
            route-filter 0.0.0.0/0 exact;
            condition prefix_11;
        }
        then accept;
    }
    term others {
        then reject;
    }
}
condition prefix_11 {
    if-route-exists {
        172.16.11.1/32;
        table inet.0;
    }
}
```

The logic of the conditional export policy can be summarized as follows: If 0/0 is present, and if 172.16.11.1/32 is present, then send the 0/0 prefix. This implies that if 172.16.11.1/32 is not present, then do not send 0/0.

The second requirement is met with an import policy on South:

```
user@South# show policy-options
policy-statement import-selected-routes {
    term 1 {
        from {
            rib inet.0;
            neighbor 10.0.78.14;
            route-filter 0.0.0.0/0 exact;
```

```
            route-filter 172.16.11.1/32 exact;
        }
        then accept;
    }
    term 2 {
        then reject;
    }
}
```

In this example, four routes are dropped as a result of the import policy on South. This is because the export policy on North leaks all of the routes received from Internet, and the import policy on South excludes some of these routes.

It is important to understand that in Junos OS, although an import policy (inbound route filter) might reject a route, not use it for traffic forwarding, and not include it in an advertisement to other peers, the router retains these routes as hidden routes. These hidden routes are not available for policy or routing purposes. However, they do occupy memory space on the router. A service provider filtering routes to control the amount of information being kept in memory and processed by a router might want the router to entirely drop the routes being rejected by the import policy.

Hidden routes can be viewed by using the `show route receive-protocol bgp` *`neighbor-address`* `hidden` command. The hidden routes can then be retained or dropped from the routing table by configuring the `keep all | none` statement at the `[edit protocols bgp]` or `[edit protocols bgp group` *`group-name`*`]` hierarchy level.

The rules of BGP route retention are as follows:

- By default, all routes learned from BGP are retained, except those where the AS path is looped. (The AS path includes the local AS.)
- By configuring the `keep all` statement, all routes learned from BGP are retained, even those with the local AS in the AS path.
- By configuring the `keep none` statement, BGP discards routes that were received from a peer and that were rejected by import policy or other sanity checking. When this statement is configured and the inbound policy changes, Junos OS re-advertises all the routes advertised by the peer.

When you configure `keep all` or `keep none` and the peers support route refresh, the local speaker sends a refresh message and performs an import evaluation. For these peers, the sessions do not restart. To determine if a peer supports refresh, check for `Peer supports Refresh capability` in the output of the `show bgp neighbor` command.

CAUTION: If you configure `keep all` or `keep none` and the peer does not support session restart, the associated BGP sessions are restarted (flapped).

Topology

Configuration

IN THIS SECTION

- CLI Quick Configuration | 695
- Configuring Conditional Installation of Prefixes | 697
- Results | 701

CLI Quick Configuration

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

Router Internet

```
set interfaces lo0 unit 0 family inet address 172.16.11.1/32
set interfaces lo0 unit 0 family inet address 172.16.12.1/32
set interfaces lo0 unit 0 family inet address 172.16.13.1/32
set interfaces lo0 unit 0 family inet address 172.16.14.1/32
set interfaces lo0 unit 0 family inet address 172.16.15.1/32
set interfaces lo0 unit 0 family inet address 192.168.9.1/32
set interfaces fe-0/1/3 unit 0 family inet address 10.0.89.14/30
set protocols bgp group toNorth local-address 10.0.89.14
set protocols bgp group toNorth peer-as 65200
set protocols bgp group toNorth neighbor 10.0.89.13
set protocols bgp group toNorth export into-bgp
set policy-options policy-statement into-bgp term 1 from interface lo0.0
set policy-options policy-statement into-bgp term 1 then accept
set routing-options router-id 192.168.9.1
set routing-options autonomous-system 65300
```

Router North

```
set interfaces fe-1/3/1 unit 0 family inet address 10.0.78.14/30
set interfaces fe-1/3/0 unit 0 family inet address 10.0.89.13/30
set interfaces lo0 unit 0 family inet address 192.168.8.1/32
set protocols bgp group toInternet local-address 10.0.89.13
set protocols bgp group toInternet peer-as 65300
set protocols bgp group toInternet neighbor 10.0.89.14
set protocols bgp group toSouth local-address 10.0.78.14
set protocols bgp group toSouth export conditional-export-bgp
set protocols bgp group toSouth peer-as 65100
set protocols bgp group toSouth neighbor 10.0.78.13
set policy-options policy-statement conditional-export-bgp term prefix_11 from protocol bgp
set policy-options policy-statement conditional-export-bgp term prefix_11 from route-filter
172.16.0.0/16 orlonger
set policy-options policy-statement conditional-export-bgp term prefix_11 then accept
set policy-options policy-statement conditional-export-bgp term conditional-default from route-
filter 0.0.0.0/0 exact
set policy-options policy-statement conditional-export-bgp term conditional-default from
condition prefix_11
set policy-options policy-statement conditional-export-bgp term conditional-default then accept
set policy-options policy-statement conditional-export-bgp term others then reject
set policy-options condition prefix_11 if-route-exists 172.16.11.1/32
set policy-options condition prefix_11 if-route-exists table inet.0
set routing-options static route 0/0 reject
set routing-options router-id 192.168.8.1
set routing-options autonomous-system 65200
```

Router South

```
set interfaces fe-0/1/2 unit 0 family inet address 10.0.78.13/30
set interfaces lo0 unit 0 family inet address 192.168.7.1/32
set protocols bgp group toNorth local-address 10.0.78.13
set protocols bgp group toNorth import import-selected-routes
set protocols bgp group toNorth peer-as 65200
set protocols bgp group toNorth neighbor 10.0.78.14
set policy-options policy-statement import-selected-routes term 1 from neighbor 10.0.78.14
set policy-options policy-statement import-selected-routes term 1 from route-filter
172.16.11.1/32 exact
set policy-options policy-statement import-selected-routes term 1 from route-filter 0.0.0.0/0
exact
```

```
set policy-options policy-statement import-selected-routes term 1 then accept
set policy-options policy-statement import-selected-routes term 2 then reject
set routing-options router-id 192.168.7.1
set routing-options autonomous-system 65100
```

Configuring Conditional Installation of Prefixes

Step-by-Step Procedure

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

To configure conditional installation of prefixes:

1. Configure the router interfaces forming the links between the three routers.

```
Router Internet
[edit interfaces]
user@Internet# set fe-0/1/3 unit 0 family inet address 10.0.89.14/30
```

```
Router North
[edit interfaces]
user@North# set fe-1/3/1 unit 0 family inet address 10.0.78.14/30
user@North# set fe-1/3/0 unit 0 family inet address 10.0.89.13/30
```

```
Router South
[edit interfaces]
user@South# set fe-0/1/2 unit 0 family inet address 10.0.78.13/30
```

2. Configure five loopback interface addresses on Router Internet to emulate BGP routes learned from the Internet that are to be imported into the routing table of Router South, and configure an additional address (192.168.9.1/32) that will be configured as the router ID.

```
Router Internet
[edit interfaces lo0 unit 0 family inet]
user@Internet# set address 172.16.11.1/32
user@Internet# set address 172.16.12.1/32
```

```
user@Internet# set address 172.16.13.1/32
user@Internet# set address 172.16.14.1/32
user@Internet# set address 172.16.15.1/32
user@Internet# set address 192.168.9.1/32
```

Also, configure the loopback interface addresses on Routers North and South.

```
Router North
[edit interfaces lo0 unit 0 family inet]
user@North# set address 192.168.8.1/32
```

```
Router South
[edit interfaces lo0 unit 0 family inet]
user@South# set address 192.168.7.1/32
```

3. Configure the static default route on Router North to be advertised to Router South.

```
[edit routing-options]
user@North# set static route 0/0 reject
```

4. Define the condition for exporting prefixes from the routing table on Router North.

```
[edit policy-options condition prefix_11]
user@North# set if-route-exists 172.16.11.1/32
user@North# set if-route-exists table inet.0
```

5. Define export policies (`into-bgp` and `conditional-export-bgp`) on Routers Internet and North respectively, to advertise routes to BGP.

> **NOTE**: Ensure that you reference the condition, `prefix_11` (configured in Step "4" on page 698), in the export policy.

```
Router Internet
[edit policy-options policy-statement into-bgp ]
```

```
user@Internet# set term 1 from interface lo0.0
user@Internet# set term 1 then accept
```

```
Router North
[edit policy-options policy-statement conditional-export-bgp]
user@North# set  term prefix_11 from protocol bgp
user@North# set  term prefix_11 from route-filter 172,16.0.0/16 orlonger
user@North# set  term prefix_11 then accept
user@North# set  term conditional-default from route-filter 0.0.0.0/0 exact
user@North# set  term conditional-default from condition prefix_11
user@North# set  term conditional-default then accept
user@North# set term others then reject
```

6. Define an import policy (`import-selected-routes`) on Router South to import some of the routes advertised by Router North into its routing table.

```
[edit policy-options policy-statement import-selected-routes ]
user@South# set term 1 from neighbor 10.0.78.14
user@South# set term 1 from route-filter 172.16.11.1/32 exact
user@South# set term 1 from route-filter 0.0.0.0/0 exact
user@South# set term 1 then accept
user@South# set term 2 then reject
```

7. Configure BGP on all three routers to enable the flow of prefixes between the autonomous systems.

NOTE: Ensure that you apply the defined import and export policies to the respective BGP groups for prefix advertisement to take place.

```
Router Internet
[edit protocols bgp group toNorth]
user@Internet# set local-address 10.0.89.14
user@Internet# set peer-as 65200
```

```
user@Internet# set neighbor 10.0.89.13
user@Internet# set export into-bgp
```

```
Router North
[edit protocols bgp group toInternet]
user@North# set local-address 10.0.89.13
user@North# set peer-as 65300
user@North# set neighbor 10.0.89.14
```

```
[edit protocols bgp group toSouth]
user@North# set local-address 10.0.78.14
user@North# set peer-as 65100
user@North# set neighbor 10.0.78.13
user@North# set export conditional-export-bgp
```

```
Router South
[edit protocols bgp group toNorth]
user@South# set local-address 10.0.78.13
user@South# set peer-as 65200
user@South# set neighbor 10.0.78.14
user@South# set import import-selected-routes
```

8. Configure the router ID and autonomous system number for all three routers.

> **NOTE**: In this example, the router ID is configured based on the IP address configured on the lo0.0 interface of the router.

```
Router Internet
[edit routing options]
user@Internet# set router-id 192.168.9.1
user@Internet# set autonomous-system 65300
```

```
Router North
[edit routing options]
```

```
user@North# set router-id 192.168.8.1
user@North# set autonomous-system 65200
```

```
Router South
[edit routing options]
user@South# set router-id 192.168.7.1
user@South# set autonomous-system 65100
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show protocols bgp`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device Internet

```
user@Internet# show interfaces
fe-0/1/3 {
    unit 0 {
        family inet {
            address 10.0.89.14/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 172.16.11.1/32;
            address 172.16.12.1/32;
            address 172.16.13.1/32;
            address 172.16.14.1/32;
            address 172.16.15.1/32;
            address 192.168.9.1/32;
        }
    }
}
```

```
user@Internet# show protocols bgp
group toNorth {
```

```
    local-address 10.0.89.14;
    export into-bgp;
    peer-as 65200;
    neighbor 10.0.89.13;
}
```

```
user@Internet# show policy-options
policy-statement into-bgp {
    term 1 {
        from interface lo0.0;
        then accept;
    }
}
```

```
user@Internet# show routing-options
router-id 192.168.9.1;
autonomous-system 65300;
```

Device North

```
user@North# show interfaces
fe-1/3/1 {
    unit 0 {
        family inet {
            address 10.0.78.14/30;
        }
    }
}
fe-1/3/0 {
    unit 0 {
        family inet {
            address 10.0.89.13/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.8.1/32;
```

```
        }
    }
}
```

```
user@North# show protocols bgp
group toInternet {
    local-address 10.0.89.13;
    peer-as 65300;
    neighbor 10.0.89.14;
}
group toSouth {
    local-address 10.0.78.14;
    export conditional-export-bgp;
    peer-as 65100;
    neighbor 10.0.78.13;
}
```

```
user@North# show policy-options
policy-statement conditional-export-bgp {
    term prefix_11 {
        from {
            protocol bgp;
            route-filter 172.16.0.0/16 orlonger;
        }
        then accept;
    }
    term conditional-default {
        from {
            route-filter 0.0.0.0/0 exact;
            condition prefix_11;
        }
        then accept;
    }
    term others {
        then reject;
    }
}
condition prefix_11 {
    if-route-exists {
        172.16.11.1/32;
```

```
        table inet.0;
    }
}
```

```
user@North# show routing-options
static {
    route 0.0.0.0/0 reject;
}
router-id 192.168.8.1;
autonomous-system 65200;
```

Device South

```
user@South# show interfaces
fe-0/1/2 {
    unit 0 {
        family inet {
            address 10.0.78.13/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.7.1/32;
        }
    }
}
```

```
user@South# show protocols bgp
bgp {
    group toNorth {
        local-address 10.0.78.13;
        import import-selected-routes;
        peer-as 65200;
        neighbor 10.0.78.14;
```

```
    }
}
```

```
user@South# show policy-options
policy-statement import-selected-routes {
    term 1 {
        from {
            neighbor 10.0.78.14;
            route-filter 172.16.11.1 exact;
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term 2 {
        then reject;
    }
}
```

```
user@South# show routing-options
router-id 192.168.7.1;
autonomous-system 65100;
```

If you are done configuring the routers, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying BGP | **706**
- Verifying Prefix Advertisement from Router Internet to Router North | **708**
- Verifying Prefix Advertisement from Router North to Router South | **709**
- Verifying BGP Import Policy for Installation of Prefixes | **710**
- Verifying Conditional Export from Router North to Router South | **711**
- Verifying the Presence of Routes Hidden by Policy (Optional) | **712**

Confirm that the configuration is working properly.

Verifying BGP

Purpose

Verify that BGP sessions have been established between the three routers.

Action

From operational mode, run the `show bgp neighbor` *neighbor-address* command.

1. Check the BGP session on Router Internet to verify that Router North is a neighbor.

```
user@Internet> show bgp neighbor 10.0.89.13
Peer: 10.0.89.13+179 AS 65200    Local: 10.0.89.14+56187 AS 65300
  Type: External    State: Established    Flags: [ImportEval Sync]
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Export: [ into-bgp ]
  Options: [Preference LocalAddress PeerAS Refresh]
  Local Address: 10.0.89.14 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 192.168.8.1     Local ID: 192.168.9.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 0
  BFD: disabled, down
  Local Interface: fe-0/1/3.0
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-unicast
  NLRI of received end-of-rib markers: inet-unicast
  NLRI of all end-of-rib markers sent: inet-unicast
  Peer supports 4 byte AS extension (peer-as 65200)
  Peer does not support Addpath
  Table inet.0 Bit: 10000
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              0
    Received prefixes:            0
    Accepted prefixes:            0
```

```
    Suppressed due to damping:    0
    Advertised prefixes:          6
  Last traffic (seconds): Received 9    Sent 18   Checked 28
  Input messages:  Total 12     Updates 1       Refreshes 0     Octets 232
  Output messages: Total 14     Updates 1       Refreshes 0     Octets 383
  Output Queue[0]: 0
```

2. Check the BGP session on Router North to verify that Router Internet is a neighbor.

```
user@North> show bgp neighbor 10.0.89.14
Peer: 10.0.89.14+56187 AS 65300  Local: 10.0.89.13+179 AS 65200
  Type: External    State: Established    Flags: [ImportEval Sync]
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Options: [Preference LocalAddress PeerAS Refresh]
  Local Address: 10.0.89.13 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 192.168.9.1     Local ID: 192.168.8.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 0
  BFD: disabled, down
  Local Interface: fe-1/3/0.0
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-unicast
  NLRI of received end-of-rib markers: inet-unicast
  NLRI of all end-of-rib markers sent: inet-unicast
  Peer supports 4 byte AS extension (peer-as 65300)
  Peer does not support Addpath
  Table inet.0 Bit: 10001
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              6
    Received prefixes:            6
    Accepted prefixes:            6
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 14   Sent 3    Checked 3
  Input messages:  Total 16     Updates 2       Refreshes 0     Octets 402
```

```
  Output messages: Total 15     Updates 0       Refreshes 0     Octets 348
  Output Queue[0]: 0
```

Check the following fields in these outputs to verify that BGP sessions have been established:

- **Peer**—Check if the peer AS number is listed.
- **Local**—Check if the local AS number is listed.
- **State**—Ensure that the value is **Established**. If not, check the configuration again and see `show bgp neighbor` for more details on the output fields.

Similarly, verify that Routers North and South form peer relationships with each other.

Meaning

BGP sessions are established between the three routers.

Verifying Prefix Advertisement from Router Internet to Router North

Purpose

Verify that the routes sent from Router Internet are received by Router North.

Action

1. From operational mode on Router Internet, run the `show route advertising-protocol bgp` *neighbor-address* command.

```
user@Internet> show route advertising-protocol bgp 10.0.89.13
inet.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.11.1/32            Self                                    I
* 172.16.12.1/32            Self                                    I
* 172.16.13.1/32            Self                                    I
* 172.16.14.1/32            Self                                    I
* 172.16.15.1/32            Self                                    I
* 192.168.9.1/32            Self                                    I
```

   The output verifies that Router Internet advertises the routes 172.16.11.1/32, 172.16.12.1/32, 172.16.13.1/32, 172.16.14.1/32, 172.16.15.1/32, and 192.168.9.1/32 (the loopback address used as router ID) to Router North.

2. From operational mode on Router North, run the `show route receive-protocol bgp` *neighbor-address* command.

```
user@North> show route receive-protocol bgp 10.0.89.14
inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.11.1/32            10.0.89.14                              65300 I
* 172.16.12.1/32            10.0.89.14                              65300 I
* 172.16.13.1/32            10.0.89.14                              65300 I
* 172.16.14.1/32            10.0.89.14                              65300 I
* 172.16.15.1/32            10.0.89.14                              65300 I
* 192.168.9.1/32            10.0.89.14                              65300 I
```

The output verifies that Router North has received all the routes advertised by Router Internet.

Meaning

Prefixes sent by Router Internet have been successfully installed into the routing table on Router North.

Verifying Prefix Advertisement from Router North to Router South

Purpose

Verify that the routes received from Router Internet and the static default route are advertised by Router North to Router South.

Action

1. From operational mode on Router North, run the `show route 0/0 exact` command.

```
user@North> show route 0/0 exact
inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:10:22
                       Reject
```

The output verifies the presence of the static default route (0.0.0.0/0) in the routing table on Router North.

2. From operational mode on Router North, run the `show route advertising-protocol bgp` *neighbor-address* command.

```
user@North> show route advertising-protocol bgp 10.0.78.13
inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               Self                                    I
* 172.16.11.1/32             Self                                 65300 I
* 172.16.12.1/32             Self                                 65300 I
* 172.16.13.1/32             Self                                 65300 I
* 172.16.14.1/32             Self                                 65300 I
* 172.16.15.1/32             Self                                 65300 I
```

   The output verifies that Router North is advertising the static route and the 172.16.11.1/32 route received from Router Internet, as well as many other routes, to Router South.

Verifying BGP Import Policy for Installation of Prefixes

Purpose

Verify that the BGP import policy successfully installs the required prefixes.

Action

See if the import policy on Router South is operational by checking if only the static default route from Router North and the 172.16.11.1/32 route from Router South are installed in the routing table.

From operational mode, run the `show route receive-protocol bgp` *neighbor-address* command.

```
user@South> show route receive-protocol bgp 10.0.78.14
inet.0: 10 destinations, 11 routes (6 active, 0 holddown, 4 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               10.0.78.14                              65200 I
* 172.16.11.1/32          10.0.78.14                              65200 65300 I
```

The output verifies that the BGP import policy is operational on Router South, and only the static default route of 0.0.0.0/0 from Router North and the 172.16.11.1/32 route from Router Internet have leaked into the routing table on Router South.

Meaning

The installation of prefixes is successful because of the configured BGP import policy.

Verifying Conditional Export from Router North to Router South

Purpose

Verify that when Device Internet stops sending the 172.16.11.1/32 route, Device North stops sending the default 0/0 route.

Action

1. Cause Device Internet to stop sending the 172.16.11.1/32 route by deactivating the 172.16.11.1/32 address on the loopback interface.

```
[edit interfaces lo0 unit 0 family inet]
user@Internet# deactivate address 172.16.11.1/32
user@Internet# commit
```

2. From operational mode on Router North, run the `show route advertising-protocol bgp` *neighbor-address* command.

```
user@North> show route advertising-protocol bgp 10.0.78.13
inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.12.1/32            Self                                   65300 I
* 172.16.13.1/32            Self                                   65300 I
* 172.16.14.1/32            Self                                   65300 I
* 172.16.15.1/32            Self                                   65300 I
```

   The output verifies that Router North is not advertising the default route to Router South. This is the expected behavior when the 172.16.11.1/32 route is not present.

3. Reactivate the 172.16.11.1/32 address on Device Internet's loopback interface.

```
[edit interfaces lo0 unit 0 family inet]
user@Internet# activate address 172.16.11.1/32
user@Internet# commit
```

Verifying the Presence of Routes Hidden by Policy (Optional)

Purpose

Verify the presence of routes hidden by the import policy configured on Router South.

> **NOTE**: This section demonstrates the effects of various changes you can make to the configuration depending on your needs.

Action

View routes hidden from the routing table of Router South by:

- Using the `hidden` option for the `show route receive-protocol bgp` *`neighbor-address`* command.
- Deactivating the import policy.

1. From operational mode, run the `show route receive-protocol bgp` *`neighbor-address`* `hidden` command to view hidden routes.

```
user@South> show route receive-protocol bgp 10.0.78.14 hidden
inet.0: 10 destinations, 11 routes (6 active, 0 holddown, 4 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  172.16.12.1/32             10.0.78.14                           65200 65300 I
  172.16.13.1/32             10.0.78.14                           65200 65300 I
  172.16.14.1/32             10.0.78.14                           65200 65300 I
  172.16.15.1/32             10.0.78.14                           65200 65300 I
```

   The output verifies the presence of routes hidden by the import policy (172.16.12.1/32, 172.16.13.1/32, 172.16.14.1/32, and 172.16.15.1/32) on Router South.

2. Deactivate the BGP import policy by configuring the `deactivate import` statement at the `[edit protocols bgp group` *`group-name`*`]` hierarchy level.

```
[edit protocols bgp group toNorth]
user@South# deactivate import
user@South# commit
```

3. Run the `show route receive-protocol bgp` *neighbor-address* operational mode command to check the routes after deactivating the import policy.

```
user@South> show route receive-protocol bgp 10.0.78.14
inet.0: 10 destinations, 11 routes (10 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               10.0.78.14                              65200 I
* 172.16.11.1/32          10.0.78.14                              65200 65300 I
* 172.16.12.1/32          10.0.78.14                              65200 65300 I
* 172.16.13.1/32          10.0.78.14                              65200 65300 I
* 172.16.14.1/32          10.0.78.14                              65200 65300 I
* 172.16.15.1/32          10.0.78.14                              65200 65300 I
```

   The output verifies the presence of previously hidden routes (172.16.12.1/32, 172.16.13.1/32, 172.16.14.1/32, and 172.16.15.1/32).

4. Activate the BGP import policy and remove the hidden routes from the routing table by configuring the `activate import` and `keep none` statements respectively at the `[edit protocols bgp group` *group-name*`]` hierarchy level.

```
[edit protocols bgp group toNorth]
user@South# activate import
user@South# set keep none
user@South# commit
```

5. From operational mode, run the `show route receive-protocol bgp` *neighbor-address* `hidden` command to check the routes after activating the import policy and configuring the `keep none` statement.

```
user@South> show route receive-protocol bgp 10.0.78.14 hidden

inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
```

   The output verifies that the hidden routes are not maintained in the routing table because of the configured `keep none` statement.

RELATED DOCUMENTATION

Conditional Advertisement Enabling Conditional Installation of Prefixes Use Cases

Conditional Advertisement and Import Policy (Routing Table) with certain match conditions

CHAPTER 10

Protecting Against DoS Attacks by Forwarding Traffic to the Discard Interface

IN THIS CHAPTER

- Assigning Forwarding Classes and Loss Priority | 714
- Understanding Forwarding Packets to the Discard Interface | 716
- Example: Forwarding Packets to the Discard Interface | 718

Assigning Forwarding Classes and Loss Priority

You can configure firewall filters to assign packet loss priority (PLP) and forwarding classes so that if congestion occurs, the marked packets can be dropped according to the priority you set. The valid match conditions are one or more of the six packet header fields: destination address, source address, IP protocol, source port, destination port, and DSCP. In other words, you can set the forwarding class and the PLP for each packet entering or an interface with a specific destination address, source address, IP protocol, source port, destination port, or DSCP.

NOTE: Junos OS assigns forwarding classes and PLP on ingress only. Do not use a filter that assigns forwarding classes or PLP as an egress filter.

When tricolor marking is enabled, a switch supports four PLP designations: `low`, `medium-low`, `medium-high`, and `high`. You can also specify any of the forwarding classes listed in Table 25 on page 714

Table 25: Unicast Forwarding Classes

Unicast Forwarding Class	For CoS Traffic Type
be	Best-effort traffic

Table 25: Unicast Forwarding Classes *(Continued)*

Unicast Forwarding Class	For CoS Traffic Type
no-loss	Guaranteed delivery for TCP traffic
fcoe	Guaranteed delivery for Fibre Channel over Ethernet (FCoE) traffic
nc	Network-control traffic

To assign forwarding classes in firewall filters:

1. Configure the family address type and filter name:

```
[edit]
user@switch# edit firewall family inetfilter ingress-filter
```

2. Configure the terms of the filter as appropriate, including the forwarding-class and loss-priority action modifiers. For example, each of the following terms in the filter examines various packet header fields and assigns the appropriate forwarding class and packet loss priority:

   - The term corp-traffic matches all IPv4 packets with a 10.1.1.0/24 source address and assigns the packets to forwarding class no-loss with a loss priority of low:

```
[edit firewall family inet filter ingress-filter]
user@switch# set term corp-traffic from source-address 10.1.1.0/24;
user@switch# set term corp-traffic then forwarding-class no-loss
user@switch# set term corp-traffic then loss-priority low
```

   - The term data-traffic matches all IPv4 packets with a 10.1.2.0/24 source address and assigns the packets to forwarding class be (best effort) with a loss priority of medium-high:

```
[edit firewall family inet filter ingress-filter]
user@switch# set term data-traffic from source-address 10.1.2.0/24;
user@switch# set term data-traffic then forwarding-class be
user@switch# set term data-traffic then loss-priority medium-high
```

- The last term accept-traffic matches any packets that did not match on any of the preceding terms and assigns the packets to forwarding class be with a loss priority of high:

```
[edit firewall family inet filter ingress-filter]
user@switch# set term accept-traffic then forwarding-class be
user@switch# set term accept-traffic then loss-priority high
```

3. Apply the filter ingress-filter to a Layer 3 interface. For information about applying the filter, see "Configuring Firewall Filters" on page 1826. (Assigning forwarding classes and PLP is supported only on ingress filters.)

RELATED DOCUMENTATION

Monitoring Firewall Filter Traffic | 827

Overview of Policers | 2198

Understanding CoS Classifiers

Understanding CoS Forwarding Classes

Understanding Forwarding Packets to the Discard Interface

The discard (dsc) interface is a virtual interface that can silently discard forwarded packets as they are received (no ICMP message is sent). It is useful in the case of a denial-of-service (DoS) attacks. Once you know the IP address that is being targeted, you can configure a policy to forward all packets received on that interface to the discard interface, where they will be dropped. Likewise, silently discarding packets that have no valid route in the associated forwarding table can prevent the device from becoming a distributed denial-of-service (DDoS) reflector, in which a spoofed source IP address is used to trigger a flood of ICMP error messages from the device.

The dsc interface can be only be configured on unit 0 of the given physical interface, and only one dsc instance per device is supported.

Configure an input filter if, for example, you want to take an action such as logging the discard to better understand the nature of the attack.

```
[edit interfaces interface-name]
dsc {
    unit 0 {
        family inet {
```

```
            filter {
                output filter-name;
            }
        }
    }
}
```

You can configure an input policy to associate a BGP community with the discard interface. To 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;
        }
    }
}
```

RELATED DOCUMENTATION

Example: Forwarding Packets to the Discard Interface | 718

Example: Forwarding Packets to the Discard Interface

IN THIS SECTION

- Requirements | 718
- Overview | 718
- Configuration | 722
- Verification | 727

This example shows how to use discard routing to mitigate denial of service (DoS) attacks, protect vital network resources from outside attack, provide protection services for customers so that each customer can initiate its own protection, and log and track DoS attempts.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In discard routing, routers are configured with rules that disallow millions of requests in a short period of time from being sent to the same address. If too many requests are received in a short period of time, the router simply discards the requests without forwarding them. The requests are sent to a router that does not forward the packets. The problematic routes are sometimes referred to as discard routes or black-holed routes. The types of routes that should be discarded are identified as attacks to customers from peers or other customers, attacks from customers to peers or other customers, attack controllers, which are hosts providing attack instructions, and unallocated address spaces, known as bogons or invalid IP addresses.

After the attack attempt is identified, operators can put a configuration in place to mitigate the attack. One way to configure discard routing in Junos OS is to create a discard static route for each next hop used for discard routes. A discard static route uses the `discard` option.

For example:

```
user@host# show routing-options
static {
    route 192.0.2.101/32 discard;
    route 192.0.2.103/32 discard;
```

```
    route 192.0.2.105/32 discard;
}
```

```
user@host> show route protocol static terse
inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A V Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* ? 192.0.2.101/32     S   5                        Discard
* ? 192.0.2.103/32     S   5                        Discard
* ? 192.0.2.105/32     S   5                        Discard
```

Another strategy, which is the main focus of this example, is to use routing policy and the discard interface. In this approach, the discard interface contains the next hop you are assigning to the null route routes. A discard interface can have only one logical unit (unit 0), but you can configure multiple IP addresses on unit 0.

For example:

```
user@host# show interfaces dsc
unit 0 {
    family inet {
        address 192.0.2.102/32 {
            destination 192.0.2.101;
        }
        address 192.0.2.104/32 {
            destination 192.0.2.103;
        }
        address 192.0.2.106/32 {
            destination 192.0.2.105;
        }
    }
}
```

```
user@host> show interfaces terse dsc
b
Interface               Admin Link Proto    Local                 Remote
dsc                     up    up
dsc.0                   up    up   inet     192.0.2.102         --> 192.0.2.101
```

```
                                        192.0.2.104         --> 192.0.2.103
                                        192.0.2.106         --> 192.0.2.105
```

The advantage of using a discard interface instead of using discard static routes is that the discard interface allows you to configure and assign filters to the interface for counting, logging, and sampling the traffic. This is demonstrated in this example.

To actually discard packets requires a routing policy attached to the BGP sessions. To locate discard-eligible routes, you can use a route filter, an access list, or a BGP community value.

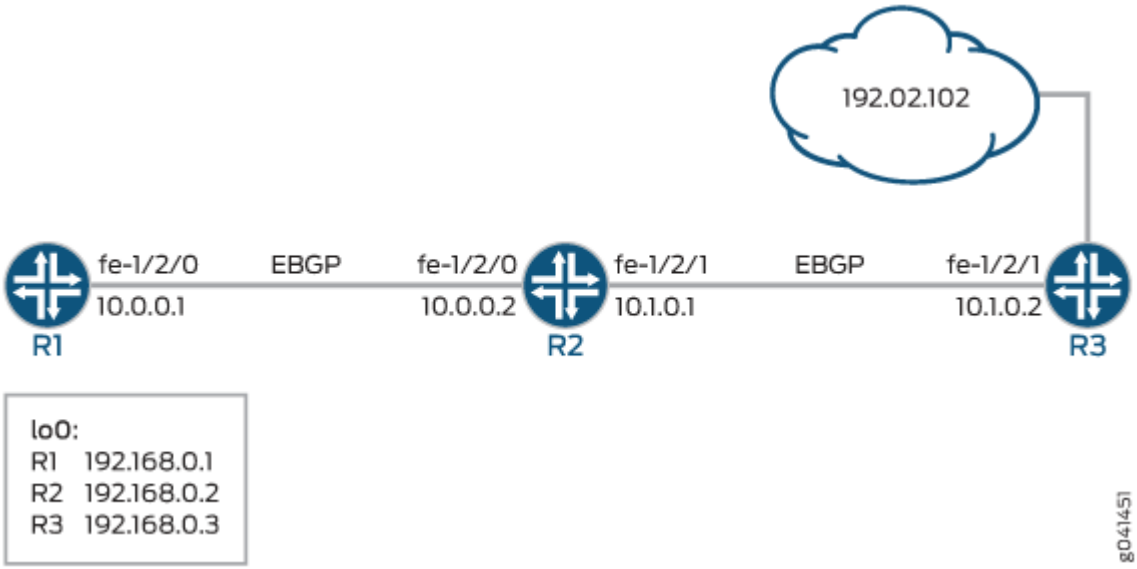
For example, here is how you would use a route filter:

Route Filter

```
protocols {
    bgp {
        import blackhole-by-route;
    }
}
policy-options {
    policy-statement blackhole-by-route {
        term specific-routes {
            from {
                route-filter 10.10.10.1/32 exact;
                route-filter 10.20.20.2/32 exact;
                route-filter 10.30.30.3/32 exact;
                route-filter 10.40.40.4/32 exact;
            }
            then {
                next-hop 192.0.2.101
            }
        }
    }
}
```

Figure 44 on page 721 shows the sample network.

Figure 44: Discard Interface Sample Network



The example includes three routers with external BGP (EBGP) sessions established.

Device R1 represents the attacking device. Device R3 represents the router closest to the device that is being attacked. Device R2 mitigates the attack by forwarding packets to the discard interface.

The example shows an outbound filter applied to the discard interface.

NOTE: An issue with using a single null route filter is visibility. All discard packets increment the same counter. To see which categories of packets are being discarded, use destination class usage (DCU), and associate a user-defined class with each null route community. Then reference the DCU classes in a firewall filter. For related examples, see "Example: Grouping Source and Destination Prefixes into a Forwarding Class" on page 671 and "Example: Configuring a Rate-Limiting Filter Based on Destination Class" on page 1208.

Compared to using route filters and access lists, using a community value is the least administratively difficult and the most scalable approach. Therefore, this is the approach shown in this example.

By default, the next hop must be equal the external BGP (EBGP) peer address. Altering the next hop for null route services requires the multihop feature to be configured on the EBGP sessions.

"CLI Quick Configuration" on page 722 shows the configuration for all of the devices in Figure 44 on page 721.

The section "No Link Title" on page 723 describes the steps on Device R2.

Configuration

IN THIS SECTION

- Procedure | 722

Procedure

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces dsc unit 0 family inet filter output log-discard
set interfaces dsc unit 0 family inet address 192.0.2.102/32 destination 192.0.2.101
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp import blackhole-policy
set protocols bgp group ext type external
set protocols bgp group ext multihop
set protocols bgp group ext export dsc-export
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
set policy-options policy-statement blackhole-policy term blackhole-communities from community
blackhole-all-routers
```

```
set policy-options policy-statement blackhole-policy term blackhole-communities then next-hop
192.0.2.101
set policy-options policy-statement dsc-export from route-filter 192.0.2.101/32 exact
set policy-options policy-statement dsc-export from route-filter 192.0.2.102/32 exact
set policy-options policy-statement dsc-export then community set blackhole-all-routers
set policy-options policy-statement dsc-export then accept
set policy-options community blackhole-all-routers members 100:5555
set routing-options static route 192.0.2.102/32 next-hop 192.0.2.101
set routing-options autonomous-system 200
set firewall filter log-discard term one then count counter
set firewall filter log-discard term one then log
```

Device R3

```
set interfaces fe-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet address 192.0.2.102/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.1.0.1
set routing-options autonomous-system 300
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R2:

1. Create the router interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set fe-1/2/1 unit 0 family inet address 10.1.0.1/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure a firewall filter that matches all packets and counts and logs the packets.

```
[edit firewall filter log-discard term one]
user@R2# set then count counter
user@R2# set then log
```

3. Create a discard interface and apply the output firewall filter.

   Input firewall filters have no impact in this context.

```
[edit interfaces dsc unit 0 family inet]
user@R2# set filter output log-discard
user@R2# set address 192.0.2.102/32 destination 192.0.2.101
```

4. Configure a static route that sends the next hop to the destination address that is specified in the discard interface.

```
[edit routing-options static]
user@R2# set route 192.0.2.102/32 next-hop 192.0.2.101
```

5. Configure BGP peering.

```
[edit protocols bgp ]
user@R2# set group ext type external
user@R2# set group ext multihop
user@R2# set group ext neighbor 10.0.0.1 peer-as 100
user@R2# set group ext neighbor 10.1.0.2 peer-as 300
```

6. Configure the routing policies.

```
[edit policy-options policy-statement blackhole-policy term blackhole-communities]
user@R2# set from community blackhole-all-routers
user@R2# set then next-hop 192.0.2.101
[edit policy-options policy-statement dsc-export]
user@R2# set from route-filter 192.0.2.101/32 exact
user@R2# set from route-filter 192.0.2.102/32 exact
user@R2# set then community set blackhole-all-routers
user@R2# set then accept
```

```
[edit policy-options community blackhole-all-routers]
user@R2# set members 100:5555
```

7. Apply the routing policies.

```
[edit protocols bgp ]
user@R2# set import blackhole-policy
user@R2# set group ext export dsc-export
```

8. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set autonomous-system 200
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show protocols` , `show policy-options`, `show routing-options`, and `show firewall` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 10.1.0.1/30;
        }
    }
}
dsc {
    unit 0 {
        family inet {
```

```
            filter {
                output log-discard;
            }
            address 192.0.2.102/32 {
                destination 192.0.2.101;
            }
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    import blackhole-policy;
    group ext {
        type external;
        multihop;
        export dsc-export;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
        neighbor 10.1.0.2 {
            peer-as 300;
        }
    }
}
```

```
user@R2# show policy-options
policy-statement blackhole-policy {
    term blackhole-communities {
        from community blackhole-all-routers;
        then {
            next-hop 192.0.2.101;
        }
```

```
    }
}
policy-statement dsc-export {
    from {
        route-filter 192.0.2.101/32 exact;
        route-filter 192.0.2.102/32 exact;
    }
    then {
        community set blackhole-all-routers;
        accept;
    }
}
community blackhole-all-routers members 100:5555;
```

```
user@R2# show routing-options
static {
    route 192.0.2.102/32 next-hop 192.0.2.101;
}
autonomous-system 200;
```

```
user@R2# show firewall
filter log-discard {
    term one {
        then {
            count counter;
            log;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Clearing the Firewall Counters | **728**
- Pinging the 192.0.2.101 Address | **728**

Checking the Output Filter | 729
Checking the Community Attribute | 730

Confirm that the configuration is working properly.

Clearing the Firewall Counters

Purpose

Clear the counters to make sure you are starting from a known zero (0) state.

Action

1. From Device R2, run the `clear firewall` command.

```
user@R2> clear firewall filter log-discard
```

2. From Device R2, run the `show firewall` command.

```
user@R2> show firewall filter log-discard
Filter: /log-discard
Counters:
Name                                                Bytes              Packets
counter                                                 0                    0
```

Pinging the 192.0.2.101 Address

Purpose

Send packets to the destination address.

Action

From Device R1, run the `ping` command.

```
user@R1> ping 192.0.2.101
PING 192.0.2.101 (192.0.2.101): 56 data bytes
^C
--- 192.0.2.101 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Meaning

As expected, the ping request fails, and no response is sent. The packets are being discarded.

Checking the Output Filter

Purpose

Verify that Device R2's firewall filter is functioning properly.

Action

From Device R2, enter the `show firewall filter log-discard` command.

```
user@R2> show firewall filter log-discard
Filter: log-discard
Counters:
Name                                                Bytes              Packets
counter                                               336                    4
```

Meaning

As expected, the counter is being incremented.

NOTE: The ping packet carries an additional 20 bytes of IP overhead as well as 8 bytes of ICMP header.

Checking the Community Attribute

Purpose

Verify that the route is being tagged with the community attribute.

Action

From Device R1, enter the `show route extensive` command, using the neighbor address for Device R2, 192.0.2.101.

```
user@R1> show route 192.0.2.101 extensive

inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
192.0.2.101/32 (1 entry, 1 announced)
TSI:
KRT in-kernel 192.0.2.101/32 -> {10.0.0.2}
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 684
                Address: 0x94141d8
                Next-hop reference count: 2
                Source: 10.0.0.2
                Next hop: 10.0.0.2 via fe-1/2/0.0, selected
                Session Id: 0x8000a
                State: <Active Ext>
                Local AS:   100 Peer AS:   200
                Age: 53:03
                Validation State: unverified
                Task: BGP_200.10.0.0.2+63097
                Announcement bits (1): 2-KRT
                AS path: 200 I
                Communities: 100:5555
                Accepted
                Localpref: 100
                Router ID: 192.168.0.2
```

Meaning

As expected, when Device R2 advertises the 192.0.2.101 route to Device R1, Device R2 adds the 100:5555 community tag.

RELATED DOCUMENTATION

Understanding Forwarding Packets to the Discard Interface | 716

Example: Configuring Routing Policy Prefix Lists | 396

CHAPTER 11

Improving Commit Times with Dynamic Routing Policies

IN THIS CHAPTER

- Understanding Dynamic Routing Policies | 732
- Example: Configuring Dynamic Routing Policies | 737

Understanding Dynamic Routing Policies

IN THIS SECTION

- Configuring Routing Policies and Policy Objects in the Dynamic Database | 733
- Configuring Routing Policies Based on Dynamic Database Configuration | 734
- Applying Dynamic Routing Policies to BGP | 735
- Preventing Reestablishment of BGP Peering Sessions After NSR Routing Engine Switchover | 736

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

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

configure and commit a routing policy based on the objects configured in the dynamic database, you can quickly update any existing routing policy by making changes to the dynamic database configuration.

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

Configuring Routing Policies and Policy Objects in the Dynamic Database

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

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

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

[edit dynamic]
user@host#
```

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

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

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

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

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

Configuring Routing Policies Based on Dynamic Database Configuration

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

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

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

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

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

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

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

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

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

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

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

Applying Dynamic Routing Policies to BGP

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

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

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

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

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

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

Preventing Reestablishment of BGP Peering Sessions After NSR Routing Engine Switchover

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

RELATED DOCUMENTATION

Example: Configuring Dynamic Routing Policies | 737

Junos OS High Availability User Guide

Example: Configuring Dynamic Routing Policies

IN THIS SECTION

- Requirements | 737
- Overview | 737
- Configuration | 739
- Verification | 750

This example shows how to configure routing policy objects in a dynamic database that is not subject to the same verification required in the standard configuration database.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

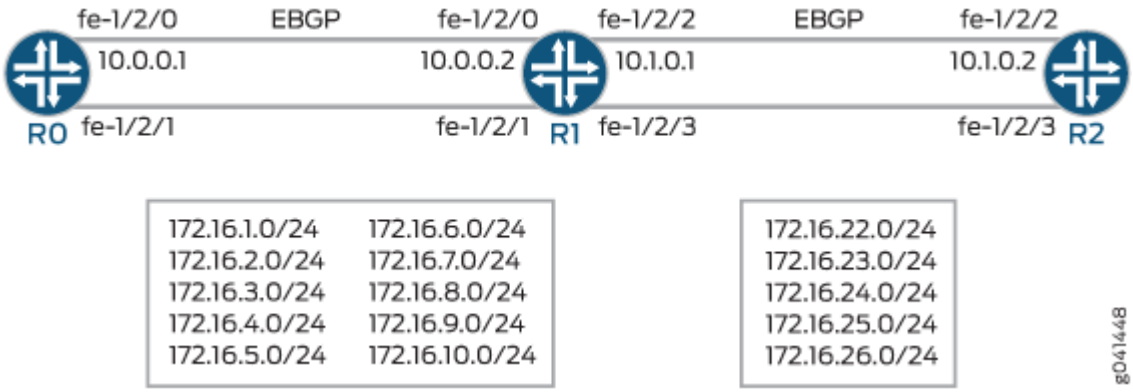
The verification process required to commit configuration changes can entail a significant amount of overhead and time.

The time it takes to commit changes to the dynamic database is much shorter than for the standard configuration database. You can 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 Junos OS does not validate configuration changes to the dynamic database, when you use this feature, you should test and verify all configuration changes before committing them.

Figure 45 on page 738 shows the sample network.

Figure 45: Dynamic Routing Policy Sample Network



The example includes three routers with external BGP (EBGP) sessions established. Only Device R1 makes use of the dynamic database.

On Device R0's fe-1/2/1 interface, multiple IPv4 interfaces are configured, and a routing policy injects these prefixes into BGP, using the `from interface fe-1/2/1.0` policy condition as a shorthand method for specifying all of the IP addresses configured on Device R0's fe-1/2/1 interface.

Likewise, on Device R2's fe-1/2/3 interface, multiple IPv4 addresses are configured, and a routing policy injects these prefixes into BGP. Device R2's configuration is slightly different from Device R0's in that Device R2's configuration demonstrates the use of a prefix list.

On Device R1, in the dynamic database, two prefix lists are defined, one for the interface addresses learned from Device R0 and another for the interface addresses learned from Device R2. Device R1's standard database contains routing policies with prefix lists that are similar to those defined in the dynamic database.

In its peer session with Device R0, Device R1 has the static-database policies applied. In contrast, in its peer session with Device R2, Device R1's configuration references the dynamic database.

The results of these different configurations are analyzed in the "Verification" on page 750 section.

"CLI Quick Configuration" on page 739 shows the configuration for all of the devices in Figure 45 on page 738.

The section "No Link Title" on page 742 describes the steps on Device R1's dynamic database.

The section "No Link Title" on page 743 describes the steps on Device R1's standard database.

Configuration

IN THIS SECTION

- Procedure | 739

Procedure

CLI Quick Configuration

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

Device R0

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.4.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.3.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.2.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.1.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.5.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.6.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.7.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.8.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.9.1/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.10.1/24
set interfaces lo0 unit 0 family inet address 10.255.14.151/32
set protocols bgp group ext type external
set protocols bgp group ext neighbor 10.0.0.2 export t2
set protocols bgp group ext neighbor 10.0.0.2 peer-as 200
set policy-options policy-statement t2 from interface fe-1/2/0.0
set policy-options policy-statement t2 from interface fe-1/2/1.0
set policy-options policy-statement t2 then accept
set routing-options router-id 10.255.14.151
set routing-options autonomous-system 100
```

Device R1 Dynamic Database

```
[edit dynamic]
set policy-options prefix-list dyn_prfx1 172.16.1.0/24
set policy-options prefix-list dyn_prfx1 172.16.2.0/24
set policy-options prefix-list dyn_prfx1 172.16.3.0/24
set policy-options prefix-list dyn_prfx1 172.16.4.0/24
set policy-options prefix-list dyn_prfx1 172.16.5.0/24
set policy-options prefix-list dyn_prfx1 172.16.6.0/24
set policy-options prefix-list dyn_prfx1 172.16.7.0/24
set policy-options prefix-list dyn_prfx1 172.16.8.0/24
set policy-options prefix-list dyn_prfx2 172.16.2.0/24
set policy-options prefix-list dyn_prfx2 172.16.3.0/24
set policy-options prefix-list dyn_prfx2 172.16.4.0/24
set policy-options prefix-list dyn_prfx2 172.16.5.0/24
set policy-options prefix-list dyn_prfx2 172.16.6.0/24
set policy-options policy-statement dyn_policy1 term t1 from prefix-list dyn_prfx1
set policy-options policy-statement dyn_policy1 term t1 then accept
set policy-options policy-statement dyn_policy1 term t2 then reject
set policy-options policy-statement dyn_policy2 term t1 from prefix-list dyn_prfx2
set policy-options policy-statement dyn_policy2 term t1 then accept
set policy-options policy-statement dyn_policy2 term t2 then reject
```

Device R1 Standard Database

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces fe-1/2/2 unit 0 family inet address 10.1.0.1/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.4.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.3.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.2.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.1.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.5.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.6.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.7.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.8.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.9.2/24
set interfaces fe-1/2/1 unit 0 family inet address 172.16.10.2/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.22.2/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.23.2/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.24.2/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.25.2/24
```

```
set interfaces fe-1/2/3 unit 0 family inet address 172.16.26.2/24
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group to_r0 idle-after-switch-over 300
set protocols bgp group to_r0 neighbor 10.0.0.1 import dyn_policy1
set protocols bgp group to_r0 neighbor 10.0.0.1 export dyn_policy2
set protocols bgp group to_r0 neighbor 10.0.0.1 peer-as 100
set protocols bgp group to_R2 import static_policy1
set protocols bgp group to_R2 export static_policy2
set protocols bgp group to_R2 idle-after-switch-over 300
set protocols bgp group to_R2 neighbor 10.1.0.2 peer-as 300
set policy-options prefix-list static_prfx1 172.16.22.0/24
set policy-options prefix-list static_prfx1 172.16.23.0/24
set policy-options prefix-list static_prfx1 172.16.24.0/24
set policy-options prefix-list static_prfx1 172.16.25.0/24
set policy-options prefix-list static_prfx2 172.16.1.0/24
set policy-options prefix-list static_prfx2 172.16.2.0/24
set policy-options prefix-list static_prfx2 172.16.3.0/24
set policy-options prefix-list static_prfx2 172.16.4.0/24
set policy-options policy-statement dyn_policy1 dynamic-db
set policy-options policy-statement dyn_policy2 dynamic-db
set policy-options policy-statement static_policy1 term t1 from prefix-list static_prfx1
set policy-options policy-statement static_policy1 term t1 then accept
set policy-options policy-statement static_policy1 term t2 then reject
set policy-options policy-statement static_policy2 term t1 from prefix-list static_prfx2
set policy-options policy-statement static_policy2 term t1 then accept
set policy-options policy-statement static_policy2 term t2 then reject
set routing-options autonomous-system 200
```

Device R2

```
set interfaces fe-1/2/2 unit 0 family inet address 10.1.0.2/30
set interfaces fe-1/2/3 unit 0 family inet address 172.16.22.1/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.23.1/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.24.1/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.25.1/24
set interfaces fe-1/2/3 unit 0 family inet address 172.16.26.1/24
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set protocols bgp group to_vin neighbor 10.1.0.1 export p1
set protocols bgp group to_vin neighbor 10.1.0.1 peer-as 200
set policy-options prefix-list ppx1 172.16.22.0/24
set policy-options prefix-list ppx1 172.16.23.0/24
set policy-options prefix-list ppx1 172.16.24.0/24
```

```
set policy-options prefix-list ppx1 172.16.25.0/24
set policy-options prefix-list ppx1 172.16.26.0/24
set policy-options policy-statement p1 term t1 from family inet
set policy-options policy-statement p1 term t1 from prefix-list ppx1
set policy-options policy-statement p1 term t1 then accept
set routing-options autonomous-system 300
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R1's dynamic database:

1. Enter configuration mode for the dynamic database.

```
user@R1> configure dynamic
Entering configuration mode
[edit dynamic]
```

2. Create a prefix list for the interface addresses learned from Device R0.

```
[edit dynamic policy-options prefix-list dyn_prfx1]
user@R1# set 172.16.1.0/24
user@R1# set 172.16.2.0/24
user@R1# set 172.16.3.0/24
user@R1# set 172.16.4.0/24
user@R1# set 172.16.5.0/24
user@R1# set 172.16.6.0/24
user@R1# set 172.16.7.0/24
user@R1# set 172.16.8.0/24
```

3. Create a prefix list for the interface addresses learned from Device R2.

```
[edit dynamic policy-options prefix-list dyn_prfx2]
user@R1# set 172.16.2.0/24
user@R1# set 172.16.3.0/24
user@R1# set 172.16.4.0/24
```

```
user@R1# set 172.16.5.0/24
user@R1# set 172.16.6.0/24
```

4. Configure the routing policies.

```
[edit dynamic policy-options policy-statement dyn_policy1]
user@R1# set term t1 from prefix-list dyn_prfx1
user@R1# set term t1 then accept
user@R1# set term t2 then reject
user@R1# set term t1 from prefix-list dyn_prfx2
user@R1# set term t1 then accept
user@R1# set term t2 then reject
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R1's standard database:

1. Create the router interfaces.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R1# set fe-1/2/2 unit 0 family inet address 10.1.0.1/30
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.4.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.3.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.2.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.1.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.5.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.6.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.7.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.8.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.9.2/24
user@R1# set fe-1/2/1 unit 0 family inet address 172.16.10.2/24
user@R1# set fe-1/2/3 unit 0 family inet address 172.16.2.2/24
user@R1# set fe-1/2/3 unit 0 family inet address 172.16.3.2/24
user@R1# set fe-1/2/3 unit 0 family inet address 172.16.4.2/24
user@R1# set fe-1/2/3 unit 0 family inet address 172.16.5.2/24
```

```
user@R1# set fe-1/2/3 unit 0 family inet address 172.16.6.2/24
user@R1# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Create routing policies that reference the policies in the dynamic database.

```
[edit policy-options]
user@R1# set policy-statement dyn_policy1 dynamic-db
user@R1# set policy-statement dyn_policy2 dynamic-db
```

3. Configure BGP peering with Device R0.

```
[edit protocols bgp group to_r0]
user@R1# set neighbor 10.0.0.1 peer-as 100
```

4. Apply the dynamic database policies to the BGP peering with Device R0.

```
[edit protocols bgp group to_r0]
user@R1# set neighbor 10.0.0.1 import dyn_policy1
user@R1# set neighbor 10.0.0.1 export dyn_policy2
```

5. Configure a prefix list for prefixes learned from Device R0.

```
[edit policy-options prefix-list static_prfx2]
user@R1# set 172.16.1.0/24
user@R1# set 172.16.2.0/24
user@R1# set 172.16.3.0/24
user@R1# set 172.16.4.0/24
```

6. Configure a prefix list for prefixes learned from Device R2.

```
[edit policy-options prefix-list static_prfx1]
user@R1# set 172.16.2.0/24
user@R1# set 172.16.3.0/24
user@R1# set 172.16.4.0/24
user@R1# set 172.16.5.0/24
```

7. Configure the static database policies.

```
[edit policy-options policy-statement static_policy1]
user@R1# set term t1 from prefix-list static_prfx1
user@R1# set term t1 then accept
user@R1# set term t2 then reject
[edit policy-options policy-statement static_policy2]
user@R1# set term t1 from prefix-list static_prfx2
user@R1# set term t1 then accept
user@R1# set term t2 then reject
```

8. Configure BGP peering with Device R2.

```
[edit protocols bgp group to_R2]
user@R1# set neighbor 10.1.0.2 peer-as 300
```

9. Apply the static database policies to the BGP peering with Device R2.

```
[edit protocols bgp group to_R2]
user@R1# set import static_policy1
user@R1# set export static_policy2
```

10. (Optional) Configure the router not to reestablish the BGP peering sessions after an active nonstop routing switchover either for a specified period or until you manually reestablish the session.

    This statement is particularly useful with dynamic routing policies because the dynamic database is not synchronized with the backup Routing Engine when nonstop active routing (NSR) is enabled. As a result, if a switchover to a backup Routing Engine occurs, import and export policies running on the primary 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.

```
[edit protocols bgp]
user@R1# set group to_r0 idle-after-switch-over 300
user@R1# set group to_R2 idle-after-switch-over 300
```

11. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R1# set routing-options autonomous-system 200
```

Results

Confirm your configuration by entering the `show` command from configuration mode in the dynamic database, and the `show interfaces`, `show protocols`, `show policy-options` and `show routing-options` commands from configuration mode in the standard database. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R1 Dynamic

```
[edit dynamic]
user@R1# show
policy-options {
    prefix-list dyn_prfx1 {
        172.16.1.0/24;
        172.16.2.0/24;
        172.16.3.0/24;
        172.16.4.0/24;
        172.16.5.0/24;
        172.16.6.0/24;
        172.16.7.0/24;
        172.16.8.0/24;
    }
    prefix-list dyn_prfx2 {
        172.16.2.0/24;
        172.16.3.0/24;
        172.16.4.0/24;
        172.16.5.0/24;
        172.16.6.0/24;
    }
    policy-statement dyn_policy1 {
        term t1 {
            from {
                prefix-list dyn_prfx1;
            }
            then accept;
        }
```

```
        term t2 {
            then reject;
        }
    }
    policy-statement dyn_policy2 {
        term t1 {
            from {
                prefix-list dyn_prfx2;
            }
            then accept;
        }
        term t2 {
            then reject;
        }
    }
}
```

Device R1 Standard

```
[edit]
user@R1# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 172.16.4.2/24;
            address 172.16.3.2/24;
            address 172.16.2.2/24;
            address 172.16.1.2/24;
            address 172.16.5.2/24;
            address 172.16.6.2/24;
            address 172.16.7.2/24;
            address 172.16.8.2/24;
            address 172.16.9.2/24;
            address 172.16.10.2/24;
        }
```

```
    }
}
fe-1/2/2 {
    unit 0 {
        family inet {
            address 10.1.0.1/30;
        }
    }
}
fe-1/2/3 {
    unit 0 {
        family inet {
            address 172.16.2.2/24;
            address 172.16.3.2/24;
            address 172.16.4.2/24;
            address 172.16.5.2/24;
            address 172.16.6.2/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    group to_r0 {
        idle-after-switch-over 300;
        neighbor 10.0.0.1 {
            import dyn_policy1;
            export dyn_policy2;
            peer-as 100;
        }
    }
    group to_R2 {
        import static_policy1;
        export static_policy2;
```

```
        idle-after-switch-over 300;
        neighbor 10.1.0.2 {
            peer-as 300;
        }
    }
}
```

```
user@R1# show policy-options
prefix-list static_prfx1 {
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
    172.16.5.0/24;
}
prefix-list static_prfx2 {
    172.16.1.0/24;
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
}
policy-statement dyn_policy1 {
    dynamic-db;
}
policy-statement dyn_policy2 {
    dynamic-db;
}
policy-statement static_policy1 {
    term t1 {
        from {
            prefix-list static_prfx1;
        }
        then accept;
    }
    term t2 {
        then reject;
    }
}
policy-statement static_policy2 {
    term t1 {
        from {
            prefix-list static_prfx2;
```

```
        }
        then accept;
    }
    term t2 {
        then reject;
    }
}
```

```
user@R1# show routing-options
autonomous-system 200;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Checking the Configured Policies on Device R1 | **750**
- Checking the Routes Advertised from Device R0 to Device R1 | **752**
- Checking the Routes That Device R1 Is Receiving from Device R0 | **753**
- Checking the Routes Advertised from Device R2 to Device R1 | **754**
- Checking the Routes That Device R1 Is Receiving from Device R2 | **754**
- Checking the Routes That Device R1 Is Advertising to Device R0 | **755**
- Checking the Routes That Device R1 Is Advertising to Device R2 | **756**

Confirm that the configuration is working properly.

Checking the Configured Policies on Device R1

Purpose

Verify that Device R1 has the dynamic and static policies in effect.

Action

From Device R1, enter the `show policy` command.

```
user@R1> show policy
Configured policies:
dyn_policy1
dyn_policy2
static_policy1
static_policy2
dyn_policy1
dyn_policy2
```

Meaning

The dynamic policies are listed two times because they are configured two times, the first and central configuration in the dynamic database. The secondary configuration is in the static database, where the dynamic database is referenced, as shown here:

Configured in the Dynamic Database

```
policy-statement dyn_policy1 {
    term t1 {
        from {
            prefix-list dyn_prfx1;
        }
        then accept;
    }
    term t2 {
        then reject;
    }
}
policy-statement dyn_policy2 {
    term t1 {
        from {
            prefix-list dyn_prfx2;
        }
        then accept;
    }
    term t2 {
        then reject;
```

```
    }
}
```

Referenced from the Static Database

```
policy-statement dyn_policy1 {
    dynamic-db;
}
policy-statement dyn_policy2 {
    dynamic-db;
}
```

Checking the Routes Advertised from Device R0 to Device R1

Purpose

Verify that Device R0's routing policy is working.

Action

From Device R0, enter the `show route advertising-protocol bgp` command, using the neighbor address for Device R1.

```
user@R0> show route advertising-protocol bgp 10.0.0.2
inet.0: 28 destinations, 28 routes (28 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.0/24             Self                                    I
* 172.16.2.0/24             Self                                    I
* 172.16.3.0/24             Self                                    I
* 172.16.4.0/24             Self                                    I
* 172.16.5.0/24             Self                                    I
* 172.16.6.0/24             Self                                    I
* 172.16.7.0/24             Self                                    I
* 172.16.8.0/24             Self                                    I
* 172.16.9.0/24             Self                                    I
* 172.16.10.0/24            Self                                    I
* 10.0.0.0/30             Self                                  I
```

Meaning

Device R0 is sending the expected routes to Device R1.

Checking the Routes That Device R1 Is Receiving from Device R0

Purpose

Verify that Device R1's import routing policy is working.

Action

From Device R1, enter the `show route receive-protocol bgp` command, using the neighbor address for Device R0.

```
user@R1> show route receive-protocol bgp 10.0.0.1
inet.0: 35 destinations, 51 routes (35 active, 0 holddown, 4 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  172.16.1.0/24              10.0.0.1                             100 I
  172.16.2.0/24              10.0.0.1                             100 I
  172.16.3.0/24              10.0.0.1                             100 I
  172.16.4.0/24              10.0.0.1                             100 I
  172.16.5.0/24              10.0.0.1                             100 I
  172.16.6.0/24              10.0.0.1                             100 I
  172.16.7.0/24              10.0.0.1                             100 I
  172.16.8.0/24              10.0.0.1                             100 I
```

Meaning

Some of the routes that are sent by Device R0 are not received by Device R1. The routes 172.16.9.0/24, 172.16.10.0/24, and 10.0.0.0/30 are missing. This is because Device R1's import policy, applied to the BGP peering session with Device R0 using the `import dyn_policy1` statement, specifically defines a prefix list limited to the following routes:

```
prefix-list dyn_prfx1 {
    172.16.1.0/24;
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
    172.16.5.0/24;
```

```
    172.16.6.0/24;
    172.16.7.0/24;
    172.16.8.0/24;
}
```

Checking the Routes Advertised from Device R2 to Device R1

Purpose

Verify that Device R2's routing policy is working.

Action

From Device R2, enter the `show route advertising-protocol bgp` command, using the neighbor address for Device R1.

```
user@R2> show route advertising-protocol bgp 10.1.0.1
inet.0: 17 destinations, 17 routes (17 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.2.0/24             Self                                    I
* 172.16.3.0/24             Self                                    I
* 172.16.4.0/24             Self                                    I
* 172.16.5.0/24             Self                                    I
* 172.16.6.0/24             Self                                    I
```

Meaning

Device R2 is sending the expected routes to Device R1.

Checking the Routes That Device R1 Is Receiving from Device R2

Purpose

Verify that Device R1's import routing policy is working.

Action

From Device R1, enter the `show route receive-protocol bgp` command, using the neighbor address for Device R0.

```
user@R1> show route receive-protocol bgp 10.1.0.2
inet.0: 35 destinations, 51 routes (35 active, 0 holddown, 4 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  172.16.2.0/24              10.1.0.2                             300 I
  172.16.3.0/24              10.1.0.2                             300 I
  172.16.4.0/24              10.1.0.2                             300 I
  172.16.5.0/24              10.1.0.2                             300 I
```

Meaning

One of the routes that is sent by Device R2 is not received by Device R1. The route 172.16.6.0/24 is missing. This is because Device R1's import policy, applied to the BGP peering session with Device R2 using the `import static_policy1` statement, specifically defines a prefix list limited to the following routes:

```
prefix-list static_prfx1 {
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
    172.16.5.0/24;
}
```

Checking the Routes That Device R1 Is Advertising to Device R0

Purpose

Verify that Device R1's export routing policy is working.

Action

From Device R1, enter the `show route advertising-protocol bgp` command, using the neighbor address for Device R0.

```
user@R1> show route advertising-protocol bgp 10.0.0.1
inet.0: 35 destinations, 51 routes (35 active, 0 holddown, 4 hidden)
```

```
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.2.0/24             Self                                    I
* 172.16.3.0/24             Self                                    I
* 172.16.4.0/24             Self                                    I
* 172.16.5.0/24             Self                                    I
* 172.16.6.0/24             Self                                    I
```

Meaning

Perhaps unexpectedly, the route that Device R1 did not receive through BGP from Device R2 (172.16.6.0/24) is nonetheless being advertised by Device R1 through BGP to Device R0. This is happening for two reasons. The first reason is that route 172.16.6.0/24 is in Device R1's routing table, albeit as a direct route, as shown here:

```
user@R1> show route 172.16.6.0/24 protocol direct
inet.0: 35 destinations, 51 routes (35 active, 0 holddown, 4 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.6.0/24          *[Direct/0] 2d 22:51:41
                      > via fe-1/2/3.0
```

The second reason is that Device R1's export policy, applied to the BGP peering session with Device R0 using the `export dyn_policy2` statement, specifically defines a prefix list limited to the following routes:

```
prefix-list dyn_prfx2 {
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
    172.16.5.0/24;
    172.16.6.0/24;
}
```

Note the inclusion of 172.16.6.0/24.

Checking the Routes That Device R1 Is Advertising to Device R2

Purpose

Verify that Device R1's export routing policy is working.

Action

From Device R1, enter the `show route advertising-protocol bgp` command, using the neighbor address for Device R2.

```
user@R1> show route advertising-protocol bgp 10.1.0.2
inet.0: 35 destinations, 51 routes (35 active, 0 holddown, 4 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 172.16.1.0/24             Self                                    I
* 172.16.2.0/24             Self                                    I
* 172.16.3.0/24             Self                                    I
* 172.16.4.0/24             Self                                    I
```

Meaning

Device R1 is sending the expected routes to Device R2. Device R1's export policy, applied to the BGP peering session with Device R2 using the `export static_policy2` statement, specifically defines a prefix list limited to the following routes:

```
prefix-list static_prfx2 {
    172.16.1.0/24;
    172.16.2.0/24;
    172.16.3.0/24;
    172.16.4.0/24;
}
```

RELATED DOCUMENTATION

Understanding Dynamic Routing Policies | 732

Example: Configuring Routing Policy Prefix Lists | 396

CHAPTER 12

Testing Before Applying Routing Policies

IN THIS CHAPTER

- Understanding Routing Policy Tests | 758
- Example: Testing a Routing Policy with Complex Regular Expressions | 760

Understanding Routing Policy Tests

IN THIS SECTION

- Example: Testing a Routing Policy | 759

Routing policy tests provide a method for verifying the effectiveness of your policies before applying them on the routing device. 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
```

Keep in mind that different protocols have different default policies that get applied if the prefix does not match the configured policy. For BGP this is accept, but for RIP it is reject. The `test policy` command always uses accept as the default policy, so unless you explicitly reject all routes that you do not want to match you might see more routes matching than you want.

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.

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
```

RELATED DOCUMENTATION

Example: Testing a Routing Policy with Complex Regular Expressions | 760

Example: Testing a Routing Policy with Complex Regular Expressions

IN THIS SECTION

- Requirements | 760
- Overview | 760
- Configuration | 763
- Verification | 769

This example shows how to test a routing policy using the `test policy` command to ensure that the policy produces the results that you expect before you apply it in a production environment. Regular expressions, especially complex ones, can be tricky to get right. This example shows how to use the `test policy` command to make sure that your regular expressions have the intended effect.

Requirements

No special configuration beyond device initialization is required before you configure this example.

Overview

IN THIS SECTION

- Topology | 762

This example shows two routing devices with an external BGP (EBGP) connection between them. Device R2 uses the BGP session to send customer routes to Device R1. These static routes have multiple community values attached.

```
user@R2> show route match-prefix 172.16.* detail

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
172.16.1.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Reject
                Address: 0x8fd0dc4
```

```
                Next-hop reference count: 8
                State: <Active Int Ext>
                Local AS: 64511
                Age: 21:32:13
                Validation State: unverified
                Task: RT
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: 64510:1 64510:10 64510:11 64510:100 64510:111

172.16.2.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Reject
                Address: 0x8fd0dc4
                Next-hop reference count: 8
                State: <Active Int Ext>
                Local AS: 64511
                Age: 21:32:13
                Validation State: unverified
                Task: RT
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: 64510:2 64510:20 64510:22 64510:200 64510:222

172.16.3.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Reject
                Address: 0x8fd0dc4
                Next-hop reference count: 8
                State: <Active Int Ext>
                Local AS: 64511
                Age: 21:32:13
                Validation State: unverified
                Task: RT
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: 64510:3 64510:30 64510:33 64510:300 64510:333

172.16.4.0/24 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Reject
                Address: 0x8fd0dc4
                Next-hop reference count: 8
```

```
                State: <Active Int Ext>
                Local AS: 64511
                Age: 21:32:13
                Validation State: unverified
                Task: RT
                Announcement bits (1): 0-KRT
                AS path: I
                Communities: 64510:4 64510:40 64510:44 64510:400 64510:444
```

To test a complex regular expression, Device R2 has a policy called `test-regex` that locates routes. The policy is configured like this:

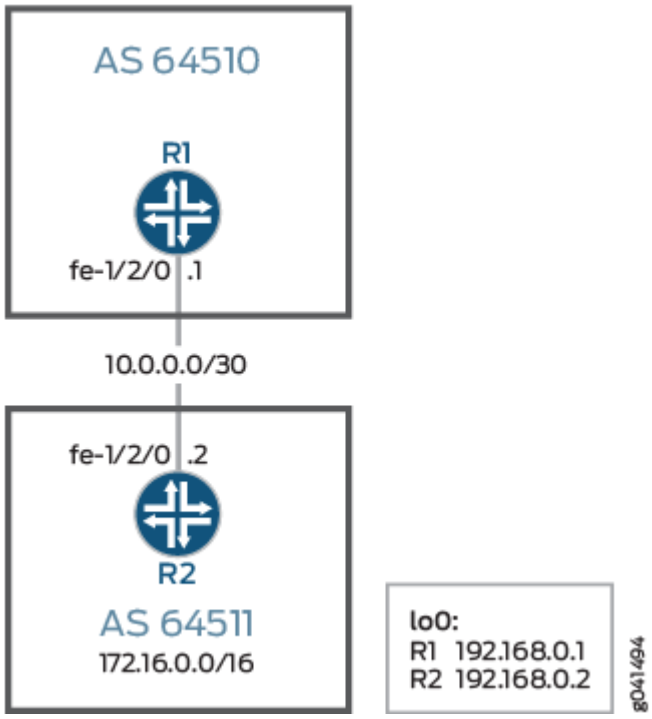
```
policy-statement test-regex {
    term find-routes {
        from community complex-regex;
        then accept;
    }
    term reject-the-rest {
        then reject;
    }
}
community complex-regex members "^64510:[13].*$";
```

This regular expression matches community values beginning with either 1 or 3.

Topology

Figure 46 on page 763 shows the sample network.

Figure 46: Routing Policy Test for Complex Regular Expressions



"CLI Quick Configuration" on page 763 shows the configuration for all of the devices in Figure 46 on page 763.

The section "No Link Title" on page 765 describes the steps on Device R2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **763**
- Procedure | **765**

CLI Quick Configuration

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

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64511
set protocols bgp group ext neighbor 10.0.0.2
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 64510
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 64510
set protocols bgp group ext neighbor 10.0.0.1
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set policy-options policy-statement send-static term 2 then reject
set policy-options policy-statement test-regex term find-routes from community complex-regex
set policy-options policy-statement test-regex term find-routes then accept
set policy-options policy-statement test-regex term reject-the-rest then reject
set policy-options community complex-regex members "^64510:[13].*$"
set routing-options static route 172.16.1.0/24 reject
set routing-options static route 172.16.1.0/24 community 64510:1
set routing-options static route 172.16.1.0/24 community 64510:10
set routing-options static route 172.16.1.0/24 community 64510:11
set routing-options static route 172.16.1.0/24 community 64510:100
set routing-options static route 172.16.1.0/24 community 64510:111
set routing-options static route 172.16.2.0/24 reject
set routing-options static route 172.16.2.0/24 community 64510:2
set routing-options static route 172.16.2.0/24 community 64510:20
set routing-options static route 172.16.2.0/24 community 64510:22
set routing-options static route 172.16.2.0/24 community 64510:200
set routing-options static route 172.16.2.0/24 community 64510:222
set routing-options static route 172.16.3.0/24 reject
set routing-options static route 172.16.3.0/24 community 64510:3
set routing-options static route 172.16.3.0/24 community 64510:30
set routing-options static route 172.16.3.0/24 community 64510:33
set routing-options static route 172.16.3.0/24 community 64510:300
```

```
set routing-options static route 172.16.3.0/24 community 64510:333
set routing-options static route 172.16.4.0/24 reject
set routing-options static route 172.16.4.0/24 community 64510:4
set routing-options static route 172.16.4.0/24 community 64510:40
set routing-options static route 172.16.4.0/24 community 64510:44
set routing-options static route 172.16.4.0/24 community 64510:400
set routing-options static route 172.16.4.0/24 community 64510:444
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 64511
```

Procedure

Step-by-Step Procedure

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

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 0 family inet address 10.0.0.2/30
user@R2# set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure BGP.

   Apply the import policy to the BGP peering session with Device R2.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set peer-as 64510
user@R2# set neighbor 10.0.0.1
```

3. Configure the routing policy that sends static routes.

```
[edit policy-options policy-statement send-static]
user@R2# set term 1 from protocol static
```

```
user@R2# set term 1 then accept
user@R2# set term 2 then reject
```

4. Configure the routing policy that tests a regular expression.

```
[edit policy-options policy-statement test-regex]
user@R2# set term find-routes from community complex-regex
user@R2# set term find-routes then accept
user@R2# set term reject-the-rest then reject
[edit policy-options community]
user@R2# set complex-regex members "^64510:[13].*$"
```

5. Configure the static routes and attaches community values.

```
[edit routing-options static route 172.16.1.0/24]
user@R2# set reject
user@R2# set community [ 64510:1 64510:10 64510:11 64510:100 64510:111 ]
[edit routing-options static route 172.16.2.0/24]
user@R2# set reject
user@R2# set community [ 64510:2 64510:20 64510:22 64510:200 64510:222 ]
[edit routing-options static route 172.16.3.0/24]
user@R2# set reject
user@R2# set community [ 64510:3 64510:30 64510:33 64510:300 64510:333 ]
[edit routing-options static route 172.16.4.0/24]
user@R2# set reject
user@R2# set community [ 64510:4 64510:40 64510:44 64510:400 64510:444 ]
```

6. Configure the autonomous system (AS) number and the router ID.

   This affects Device R2's routing table, and as no impact on Device R1 and Device R3.

```
[edit routing-options ]
user@R2# set router-id 192.168.0.2
user@R2# set autonomous-system 64511
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group ext {
        type external;
        peer-as 64510;
        neighbor 10.0.0.1;
    }
}
```

```
user@R2# show policy-options
policy-statement send-static {
    term 1 {
        from protocol static;
        then accept;
    }
    term 2 {
        then reject;
```

```
    }
}
policy-statement test-regex {
    term find-routes {
        from community complex-regex;
        then accept;
    }
    term reject-the-rest {
        then reject;
    }
}
community complex-regex members "^64510:[13].*$";
```

```
user@R2# show routing-options
static {
    route 172.16.1.0/24 {
        reject;
        community [ 64510:1 64510:10 64510:11 64510:100 64510:111 ];
    }
    route 172.16.2.0/24 {
        reject;
        community [ 64510:2 64510:20 64510:22 64510:200 64510:222 ];
    }
    route 172.16.3.0/24 {
        reject;
        community [ 64510:3 64510:30 64510:33 64510:300 64510:333 ];
    }
    route 172.16.4.0/24 {
        reject;
        community [ 64510:4 64510:40 64510:44 64510:400 64510:444 ];
    }
}
router-id 192.168.0.2;
autonomous-system 64511;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Test to See Which Communities Match the Regular Expression | 769

Confirm that the configuration is working properly.

Test to See Which Communities Match the Regular Expression

Purpose

You can test the regular expression and its policy by using the `test policy`*policy-name* command.

Action

1. On Device R2, run the `test policy test-regex 0/0` command.

```
user@R2> test policy test-regex 0/0


inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.1.0/24      *[Static/5] 1d 00:32:50
                      Reject
172.16.3.0/24      *[Static/5] 1d 00:32:50
                      Reject

Policy test-regex: 2 prefix accepted, 5 prefix rejected
```

2. On Device R2, change the regular expression to match a community value containing any number of instances of the digit 2.

```
[edit policy-options community complex-regex]
user@R2# delete members "^64510:[13].*$"
```

```
user@R2# set members "^65020:2+$"
user@R2# commit
```

3. On Device R2, rerun the `test policy test-regex 0/0` command.

```
user@R2> test policy test-regex 0/0

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.16.2.0/24      *[Static/5] 1d 00:31:36
                      Reject

Policy test-regex: 1 prefix accepted, 6 prefix rejected
```

Meaning

The 172.16.1.0 /24 and 172.16.3.0/24 routes both have communities attached that match the ^64510:[13].*$ expression. The 172.16.2.0/24 route has communities that match the ^65020:2+$ expression.

2

PART

Configuring Firewall Filters

Understanding How Firewall Filters Protect Your Network | 772

Firewall Filter Match Conditions and Actions | 845

Applying Firewall Filters to Routing Engine Traffic | 1057

Applying Firewall Filters to Transit Traffic | 1138

Configuring Firewall Filters in Logical Systems | 1213

Configuring Firewall Filter Accounting and Logging | 1277

Attaching Multiple Firewall Filters to a Single Interface | 1300

Attaching a Single Firewall Filter to Multiple Interfaces | 1368

Configuring Filter-Based Tunneling Across IP Networks | 1394

Configuring Service Filters | 1431

Configuring Simple Filters | 1463

Configuring Layer 2 Firewall Filters | 1479

Configuring Firewall Filters for Forwarding, Fragments, and Policing | 1493

Configuring Firewall Filters (EX Series Switches) | 1534

Configuring Firewall Filters (QFX Series Switches, EX4600 Switches, PTX Series Routers) | 1715

Configuring Firewall Filter Accounting and Logging (EX9200 Switches) | 1883

CHAPTER 13

Understanding How Firewall Filters Protect Your Network

IN THIS CHAPTER

- Firewall Filters Overview | 772
- Router Data Flow Overview | 773
- Stateless Firewall Filter Overview | 776
- Understanding How to Use Standard Firewall Filters | 778
- Understanding How Firewall Filters Control Packet Flows | 779
- Stateless Firewall Filter Components | 781
- Stateless Firewall Filter Application Points | 788
- How Standard Firewall Filters Evaluate Packets | 793
- Understanding Firewall Filter Fast Lookup Filter | 798
- Understanding Egress Firewall Filters with PVLANs | 799
- Selective Class-based Filtering on PTX Routers | 800
- Guidelines for Configuring Firewall Filters | 814
- Guidelines for Applying Standard Firewall Filters | 821
- Supported Standards for Filtering | 826
- Monitoring Firewall Filter Traffic | 827
- Troubleshooting Firewall Filters | 830

Firewall Filters Overview

Firewall filters provide a means of protecting your router (and switch) from excessive traffic transiting the router (and switch) to a network destination or destined for the Routing Engine. Firewall filters that control local packets can also protect your router (and switch) from external incidents.

You can configure a *firewall filter* to do the following:

- Restrict traffic destined for the Routing Engine based on its source, protocol, and application.
- Limit the traffic rate of packets destined for the Routing Engine to protect against flood, or denial-of-service (DoS) attacks.
- Address special circumstances associated with fragmented packets destined for the Routing Engine. Because the device evaluates every packet against a firewall filter (including fragments), you must configure the filter to accommodate fragments that do not contain packet header information. Otherwise, the filter discards all but the first fragment of a fragmented packet.

RELATED DOCUMENTATION

Stateless Firewall Filter Types

Guidelines for Configuring Firewall Filters | 814

Guidelines for Applying Standard Firewall Filters | 821

Understanding How to Use Standard Firewall Filters | 778

Router Data Flow Overview

IN THIS SECTION

- Flow of Routing Information | 773
- Flow of Data Packets | 774
- Flow of Local Packets | 774
- Interdependent Flows of Routing Information and Packets | 775
- Stateless and Stateful Firewall Filters | 775

The Junos® operating system (Junos OS) provides a *policy framework*, which is a collection of Junos OS policies that enable you to control flows of routing information and packets within the router.

Flow of Routing Information

Routing information is the information about routes learned by the routing protocols from a router's neighbors. This information is stored in routing tables. 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.

To control which routes the routing protocols place in the routing tables and which routes the routing protocols advertise from the routing tables, you can configure *routing policies*, which are sets of rules that the policy framework uses to preempt default routing policies.

The Routing Engine, which runs the router's control plane software, handles the flow of routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. The Routing Engine runs the Junos OS and routing policies and stores the active router configuration, the master routing table, and the master forwarding table,

Flow of Data Packets

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.

The Packet Forwarding Engine, which is the central processing element of the router's forwarding plane, handles the flow of data packets in and out of the router's physical interfaces. Although the Packet Forwarding Engine contains Layer 3 and Layer 4 header information, it does not contain the packet data itself (the packet's payload).

To control the flow of data packets transiting the device as the packets are being forwarded from a source to a destination, you can apply stateless firewall filters to the input or output of the router's or switch's physical interfaces.

To enforce a specified bandwidth and maximum burst size for traffic sent or received on an interface, you can configure policers. Policers are a specialized type of stateless firewall filter and a primary component of the Junos OS class-of-service (CoS).

Flow of Local Packets

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.

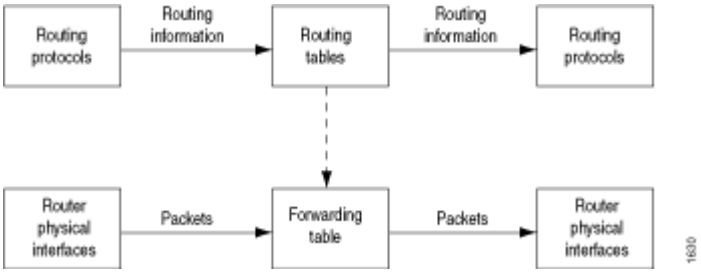
The Routing Engine handles the flow of local packets from the router's physical interfaces and to the Routing Engine.

To control the flow of local packets between the physical interfaces and the Routing Engine, you can apply stateless firewall filters to the input or output of the loopback interface. The loopback interface (lo0) is the interface to the Routing Engine and carries no data packets.

Interdependent Flows of Routing Information and Packets

Figure 47 on page 775 illustrates the flow of data through a router. Although routing information flows and packet flows are very different from one another, they are also interdependent.

Figure 47: Flows of Routing Information and Packets



Routing policies determine which routes the Routing Engine places 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.

Stateless and Stateful Firewall Filters

A stateless firewall filter, also known as an access control list (ACL), does not statefully inspect traffic. Instead, it evaluates packet contents statically and does not keep track of the state of network connections. In contrast, a stateful firewall filter uses connection state information derived from other applications and past communications in the data flow to make dynamic control decisions.

The basic purpose of a stateless firewall filter is to enhance security through the use of packet filtering. Packet filtering enables you to inspect the components of incoming or outgoing packets and then perform the actions you specify on packets that match the criteria you specify. The typical use of a stateless firewall filter is to protect the Routing Engine processes and resources from malicious or untrusted packets.

RELATED DOCUMENTATION

Stateless Firewall Filter Components | 781

Understanding Firewall Filter Processing Points for Bridged and Routed Packets | 1853

Understanding How Firewall Filters Are Evaluated | 854

Configuring Firewall Filters | 1826
Understanding Route Preference Values (Administrative Distance)
Understanding Routing Policies | 22

Stateless Firewall Filter Overview

IN THIS SECTION

- Packet Flow Control | 776
- Stateless and Stateful Firewall Filters | 777
- Purpose of Stateless Firewall Filters | 777

Packet Flow Control

To influence which packets are allowed to transit the system and to apply special actions to packets as necessary, you can configure *stateless firewall filters*. A stateless firewall specifies a sequence of one or more packet-filtering rules, called *filter terms*. A filter term specifies *match conditions* to use to determine a match and *actions* to take on a matched packet. A stateless *firewall filter* enables you to manipulate any packet of a particular protocol family, including fragmented packets, based on evaluation of Layer 3 and Layer 4 header fields. You typically apply a stateless firewall filter to one or more interfaces that have been configured with protocol family features. You can apply a stateless firewall filter to an ingress interface, an egress interface, or both.

Data Packet Flow Control

To control the flow of data packets transiting the device as the packets are being forwarded from a source to a destination, you can apply stateless firewall filters to the input or output of the router's or switch's physical interfaces.

To enforce a specified bandwidth and maximum burst size for traffic sent or received on an interface, you can configure *policers*. Policers are a specialized type of stateless firewall filter and a primary component of the Junos OS *class-of-service* (CoS).

Local Packet Flow Control

To control the flow of local packets between the physical interfaces and the Routing Engine, you can apply stateless firewall filters to the input or output of the *loopback interface*. The loopback interface (`lo0`) is the interface to the Routing Engine and carries no data packets.

Junos OS Evolved Local Packet Flow Control

In Junos OS Evolved, you can have two different filters: one for network control traffic (loopback traffic) and one for management traffic. With two filters, you have more flexibility. For example, you can configure a stricter filter on management interface traffic than on network control traffic.

Management filtering uses Routing Engine filters based on netfilters, a framework provided by the Linux kernel. This difference results in only certain matches and actions being supported. Refer Routing Engine Firewall Filters to read more.

NOTE: You must explicitly add the filter on the management interface as for Junos OS Evolved, the lo filter no longer applies on the management traffic, as is the case for Junos OS.

Stateless and Stateful Firewall Filters

A stateless firewall filter, also known as an *access control list* (ACL), does not statefully inspect traffic. Instead, it evaluates packet contents statically and does not keep track of the state of network connections. In contrast, a *stateful firewall filter* uses connection state information derived from other applications and past communications in the data flow to make dynamic control decisions.

The Routing Policies, Firewall Filters, and Traffic Policers User Guide describes *stateless firewall filters*.

Purpose of Stateless Firewall Filters

The basic purpose of a stateless firewall filter is to enhance security through the use of packet filtering. Packet filtering enables you to inspect the components of incoming or outgoing packets and then perform the actions you specify on packets that match the criteria you specify. The typical use of a stateless firewall filter is to protect the Routing Engine processes and resources from malicious or untrusted packets.

RELATED DOCUMENTATION

Router Data Flow Overview | 773

Stateless Firewall Filter Types

Controlling Network Access Using Traffic Policing Overview | 1916

Packet Flow Through the Junos OS CoS Process Overview

Understanding How to Use Standard Firewall Filters

IN THIS SECTION

- Using Standard Firewall Filters to Affect Local Packets | 778
- Using Standard Firewall Filters to Affect Data Packets | 779

Using Standard Firewall Filters to Affect Local Packets

On a router, you can configure one physical loopback interface, **lo0**, and one or more addresses on the interface. The loopback interface is the interface to the Routing Engine, which runs and monitors all the control protocols. The loopback interface carries local packets only. Standard firewall filters applied to the loopback interface affect the local packets destined for or transmitted from the Routing Engine.

NOTE: When you create an additional loopback interface, it is important to apply a filter to it so the Routing Engine is protected. We recommend that when you apply a filter to the loopback interface, you include the **apply-groups** statement. Doing so ensures that the filter is automatically inherited on every loopback interface, including **lo0** and other loopback interfaces.

Trusted Sources

The typical use of a standard stateless *firewall filter* is to protect the Routing Engine processes and resources from malicious or untrusted packets. To protect the processes and resources owned by the Routing Engine, you can use a standard stateless firewall filter that specifies which protocols and services, or applications, are allowed to reach the Routing Engine. Applying this type of filter to the loopback interface ensures that the local packets are from a trusted source and protects the processes running on the Routing Engine from an external attack.

Flood Prevention

You can create standard stateless firewall filters that limit certain TCP and ICMP traffic destined for the Routing Engine. A router without this kind of protection is vulnerable to TCP and ICMP flood attacks, which are also called denial-of-service (DoS) attacks. For example:

- A TCP flood attack of SYN packets initiating connection requests can overwhelm the device until it can no longer process legitimate connection requests, resulting in denial of service.
- An ICMP flood can overload the device with so many echo requests (ping requests) that it expends all its resources responding and can no longer process valid network traffic, also resulting in denial of service.

Applying the appropriate firewall filters to the Routing Engine protects against these types of attacks.

Using Standard Firewall Filters to Affect Data Packets

Standard firewall filters that you apply to your router's transit interfaces evaluate only the user data packets that transit the router from one interface directly to another as they are being forwarded from a source to a destination. To protect the network as a whole from unauthorized access and other threats at specific interfaces, you can apply firewall filters router transit interfaces .

RELATED DOCUMENTATION

- How Standard Firewall Filters Evaluate Packets
- Guidelines for Configuring Firewall Filters
- Guidelines for Applying Standard Firewall Filters

Understanding How Firewall Filters Control Packet Flows

A switch supports firewall filters that allow you to control flows of data packets and local packets. *Data packets* transit a switch as they are forwarded from a source to a destination. *Local packets* are destined for or sent by a Routing Engine (they do not transit a switch). Local packets usually contain routing protocol data, data for IP services such as Telnet or SSH, or data for administrative protocols such as the Internet Control Message Protocol (ICMP).

Firewall filters affect packet flows entering into or exiting from a switch as follows:

- Ingress firewall filters affect the flow of data packets that are received on switch interfaces. When a switch receives a data packet, the Packet Forwarding Engine in the system that contains the ingress interface determines where to forward the packet by looking in its Layer 2 or Layer 3 forwarding

table for the best route to the destination. Data packets are forwarded to an egress interface. Locally destined packets are forwarded to the Routing Engine.

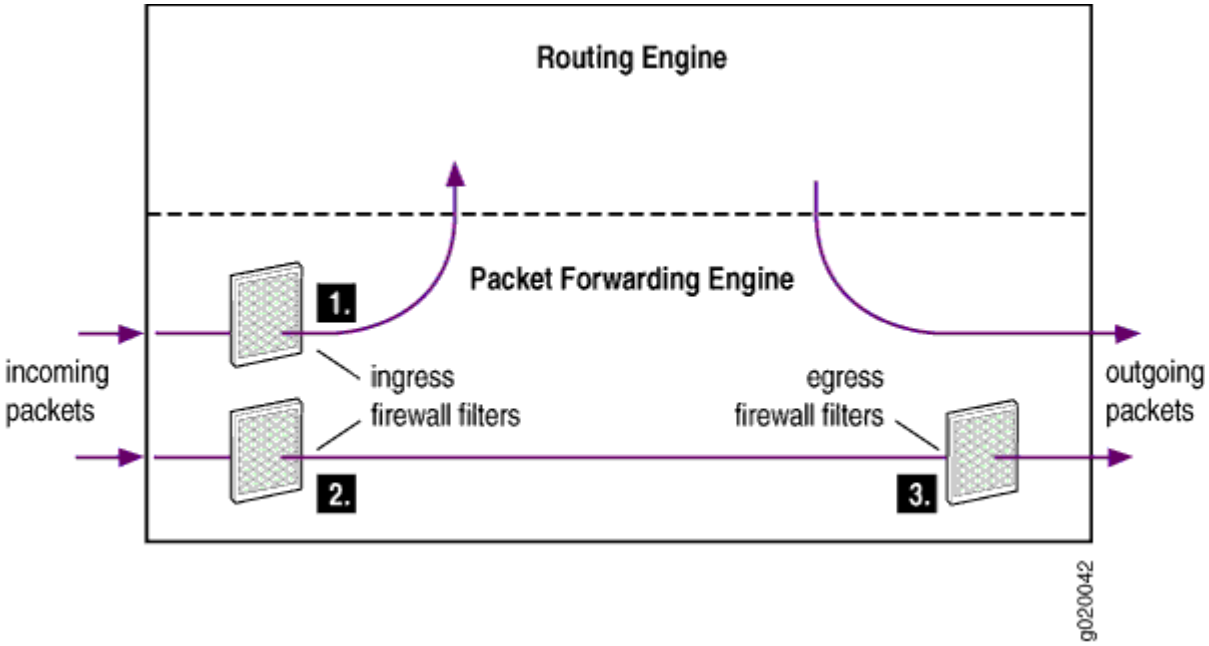
- Egress firewall filters affect data packets that are transiting a switch but do not affect packets sent by the Routing Engine. These filters are applied by the Packet Forwarding Engine in the system that contains the egress interface.

NOTE: On some QFX platforms which use PTX codebase, such as QFX10002-60C for example, there is support for firewall filter inspection of RE-based outgoing traffic.

Figure 48 on page 780 illustrates the application of ingress and egress firewall filters to control the flow of packets through a switch:

1. Ingress *firewall filter* applied to locally destined packets that are received on switch interfaces and are destined for the Routing Engine.
2. Ingress firewall filter applied to data packets that are received on switch interfaces and will transit the switch.
3. Egress firewall filter applied to data packets that are transiting the switch.

Figure 48: Application of Firewall Filters to Control Packet Flow



Stateless Firewall Filter Components

IN THIS SECTION

- Protocol Family | 781
- Filter Type | 782
- Terms | 784
- Match Conditions | 785
- Actions | 786

This topic covers the following information:

Protocol Family

Under the `firewall` statement, you can specify the protocol family for which you want to filter traffic.

Table 26 on page 781 describes the *firewall filter* protocol families.

Table 26: Firewall Filter Protocol Families

Type of Traffic to Be Filtered	Configuration Statement	Comments
Protocol Independent	`family any`	All protocol families configured on a *logical interface*.
Internet Protocol version 4 (IPv4)	`family inet`	The `family inet` statement is optional for IPv4.
Internet Protocol version 6 (IPv6)	`family inet6`	
MPLS	`family mpls`	
MPLS-tagged IPv4	`family mpls`	Supports matching on IP addresses and ports, up to five MPLS stacked labels.

Table 26: Firewall Filter Protocol Families *(Continued)*

Type of Traffic to Be Filtered	Configuration Statement	Comments
MPLS-tagged IPv6	`family mpls`	Supports matching on IP addresses and ports, up to five MPLS stacked labels.
Virtual private LAN service (VPLS)	`family vpls`	
Layer 2 Circuit Cross-Connection	`family ccc`	
Layer 2 Bridging	`family bridge` (for MX Series routers) and `family ethernet-switching` (for EX Series switches)	MX Series routers and EX Series switches only.

Filter Type

Under the `family` *`family-name`* statement, you can specify the type and name of the filter you want to configure.

Table 27 on page 783 describes the firewall filter types.

Table 27: Filter Types

Filter Type	Configuration Statement	Description
Standard Firewall Filter	`filter` *filter-name*	Filters the following traffic types: • Protocol independent • IPv4 • IPv6 • MPLS • MPLS-tagged IPv4 • MPLS-tagged IPv6 • VPLS • Layer 2 CCC • Layer 2 bridging (MX Series routers and EX Series switches only)
Service Filter	`service-filter` *service-filter-name*	Defines packet-filtering to be applied to ingress or egress before it is accepted for service processing or applied to returning service traffic after service processing has completed. Filters the following traffic types: • IPv4 • IPv6 Supported at logical interfaces configured on the following hardware only: • Adaptive Services (AS) PICs on M Series and T Series routers • Multiservices (MS) PICs on M Series and T Series routers • Multiservices (MS) DPCs on MX Series routers (and EX Series switches)

Table 27: Filter Types *(Continued)*

Filter Type	Configuration Statement	Description
Simple Filter	`simple-filter` *`simple-filter-name`*	Defines packet filtering to be applied to ingress traffic only. Filters the following traffic type: • IPv4 Supported at logical interfaces configured on the following hardware only: • Gigabit Ethernet Intelligent Queuing (IQ2) PICs installed on M120, M320, or T Series routers • Enhanced Queuing Dense Port Concentrators (EQ DPCs) installed on MX Series routers (and EX Series switches)

Terms

Under the `filter`, `service-filter`, or `simple-filter` statement, you must configure at least one firewall filter *term*. A term is a named structure in which match conditions and actions are defined. Within a firewall filter, you must configure a unique name for each term.

TIP: For each protocol family on an interface, you can apply no more than one filter in each direction. If you try to apply additional filters for the same protocol family in the same direction, the last filter overwrites the previous filter. You can, however, apply filters from the same protocol family to the input and output direction of the same interface.

All stateless firewall filters contain one or more terms, and each term consists of two components—match conditions and actions. The match conditions define the values or fields that the packet must contain to be considered a match. If a packet is a match, the corresponding action is taken. By default, a packet that does not match a firewall filter is discarded.

If a packet arrives on an interface for which no firewall filter is applied for the incoming traffic on that interface, the packet is accepted by default.

NOTE: A firewall filter with a large number of terms can adversely affect both the configuration commit time and the performance of the Routing Engine.

Additionally, you can configure a stateless firewall filter within the term of another filter. This method enables you to add common terms to multiple filters without having to modify all filter definitions. You can configure one filter with the desired common terms, and configure this filter as a term in other filters. Consequently, to make a change in these common terms, you need to modify only one filter that contains the common terms, instead of multiple filters.

Match Conditions

A firewall filter term must contain at least one packet-filtering criteria, called a *match condition*, to specify the field or value that a packet must contain in order to be considered a match for the firewall filter term. For a match to occur, the packet must match all the conditions in the term. If a packet matches a firewall filter term, the router (or switch) takes the configured action on the packet.

If a firewall filter term contains multiple match conditions, a packet must meet *all* match conditions to be considered a match for the firewall filter term.

If a single match condition is configured with multiple values, such as a range of values, a packet must match only *one* of the values to be considered a match for the firewall filter term.

The scope of match conditions you can specify in a firewall filter term depends on the protocol family under which the firewall filter is configured. You can define various match conditions, including the IP source address field, IP destination address field, TCP or 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. These are pre-defined, or fixed, match conditions.

On MX Series 3D Universal Edge Routers with MPCs or MICs, it is possible to build flexible match conditions for IPv4, IPv6, Layer 2 bridge, CCC, and VPLS protocol families. These flexible match conditions allow a user to specify start location, byte offset, match length, and other parameters within the packet.

Each protocol family supports a different set of match conditions, and some match conditions are supported only on certain routing devices. For example, a number of match conditions for VPLS traffic are supported only on the MX Series 3D Universal Edge Routers.

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 the order of the conditions in the `from` statement is not important.

If an individual match condition can specify a list of values (such as multiple source and destination addresses) or a range of numeric values, a match occurs if any of the values matches the packet.

If a filter term does not specify match conditions, the term accepts all packets and the actions specified in the term's `then` statement are optional.

NOTE: Some of the numeric range and bit-field match conditions allow you to specify a text synonym. For a complete list of synonyms:

- If you are using the J-Web interface, select the synonym from the appropriate list.
- If you are using the CLI, type a question mark (?) after the `from` statement.

Actions

The actions specified in a firewall filter term define the actions to take for any packet that matches the conditions specified in the term.

Actions that are configured within a single term are all taken on traffic that matches the conditions configured.

BEST PRACTICE: We strongly recommend that you explicitly configure one or more actions per firewall filter term. Any packet that matches all the conditions of the term is automatically accepted unless the term specifies other or additional actions.

Firewall filter actions fall into the following categories:

Filter-Terminating Actions

A filter-terminating action halts all evaluation of a firewall filter for a specific packet. The router (or switch) 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.

The presence of a nonterminating action, such as `count`, `log`, or `syslog`, without an explicit terminating action, such as `accept`, `discard`, or `reject`, results in a default terminating action of `accept`. If you do not want the firewall filter action to terminate, use the `next term` action after the nonterminating action.

NOTE: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

In this example, term 2 is never evaluated, because term 1 has the implicit default `accept` terminating action.

```
[edit firewall filter test]
term 1 {
    from {
        source-address {
            0.0.0.0/0;
        }
    }
    then {
        log;
        <accept>  #By default if not specified
    }
}
term 2 {
    then {
        reject;
    }
}
```

In this example, term 2 is evaluated, because term 1 has the explicit `next term` flow control action.

```
[edit firewall filter test]
term 1 {
    from {
        source-address {
            0.0.0.0/0;
        }
    }
    then {
        log;
        next term;
    }
}
```

```
term 2 {
    then {
        reject;
    }
}
```

Flow Control Action

For standard stateless firewall filters only, the action `next term` enables the router (or switch) to perform configured actions on the packet and then evaluate the following term in the filter, rather than terminating the filter.

A maximum of 1024 `next term` actions are supported per standard stateless firewall filter configuration. If you configure a standard filter that exceeds this limit, your candidate configuration results in a commit error.

RELATED DOCUMENTATION

- Stateless Firewall Filter Types
- Firewall Filter Flexible Match Conditions | **862**
- *Inserting a New Identifier in a Junos OS Configuration*
- Junos OS CLI User Guide

Stateless Firewall Filter Application Points

After you define the *firewall filter*, you must apply it to an application point. These application points include logical interfaces, physical interfaces, routing interfaces, and routing instances.

In most cases, 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 together multiple firewall filters (with single or multiple terms) 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 list or an output list.

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.

Table 28 on page 789 describes each point to which you can apply a firewall filter. For each application point, the table describes the types of firewall filters supported at that point, the router (or switch) hierarchy level at which the filter can be applied, and any platform-specific limitations.

Table 28: Stateless Firewall Filter Configuration and Application Summary

Filter Type	Application Point	Restrictions

Stateless firewall filter
 	
Configure by including the `filter` *`filter-name`* statement the `[edit firewall]` hierarchy level:
 	
`filter` *`filter-name`*;
 	
NOTE: If you do not include the `family` statement, the firewall filter processes IPv4 traffic by default.
	Logical interface
	
	Apply at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet]` hierarchy level by including the `input` *`filter-name`* or `output` *`filter-name`* statements:
	
	`filter {`
	`  input` *`filter-name`*;
	`  output` *`filter-name`*;
	`}`
	
	NOTE: A filter configured with the implicit `inet` protocol family cannot be included in an input filter list or an output filter list.
	
	NOTE: On T4000 Type 5 FPCs, a filter attached at the Layer 2 application point (that is, at the logical interface level) is unable to match with the forwarding class of a packet that is set by a Layer 3 classifier such as DSCP, DSCP V6, `inet-precedence`, and `mpls-exp`.
		Supported on the following routers:
		
		• T Series routers
		• M320 routers
		• M7i routers with the enhanced CFEB (C FEB-e)
		• M10i routers with the enhanced CFEB-e
		
		Also supported on the following Modular Port Concentrators (MPCs) on MX Series routers:
		
		• 10-Gigabit Ethernet MPC
		• 60-Gigabit Ethernet Queuing MPC
		• 60-Gigabit Ethernet Enhanced Queuing MPC
		• 100-Gigabit Ethernet MPC
		• Also supported on EX Series switches

Table 28: Stateless Firewall Filter Configuration and Application Summary *(Continued)*

Filter Type	Application Point	Restrictions

Stateless firewall filter
 	
Configure at the `[edit firewall family family-name]` hierarchy level by including the following statement:
 	
`filter filter-name;`
 	
The *family-name* can be any of the following protocol families:
• any
• bridge
• **ethernet-switching**
• **ccc**
• inet
• inet6
• mpls
• vpls
	Protocol family on a logical interface
	
	Apply at the `[edit interfaces interface-name unit unit-number family family-name]` hierarchy level by, including the input, input-list, output, or output-list statements:
	
	`filter {`
	`  input filter-name;`
	`  input-list [ filter-names ];`
	`  output filter-name;`
	`  output-list [ filter-names ];`
	`}`
		The protocol family bridge is supported only on MX Series routers.

Stateless firewall filter	Routing Engine loopback interface	

Table 28: Stateless Firewall Filter Configuration and Application Summary *(Continued)*

Filter Type	Application Point	Restrictions

Service filter
 	
Configure at the [edit firewall family (inet | inet6)] hierarchy level by including the following statement:
 	
service-filter *service-filter-name*;
	Family inet or inet6 on a logical interface
	
	Apply at the [edit interfaces *interface-name* unit *unit-number* family (inet | inet6)] hierarchy level by using the service-set statement to apply a service filter as an input or output filter to a service set:
	
	service {
	input {
	service-set *service-set-name* service-filter *filter-name;*
	
	}
	output {
	service-set *service-set-name* service-filter *filter-name;*
	
	}
	}
	
	Configure a service set at the [edit services] hierarchy level by including the following statement:
	
	service-set *service-set-name*;
		Supported only on Adaptive Services (AS) and Multiservices (MS) PICs.

Table 28: Stateless Firewall Filter Configuration and Application Summary *(Continued)*

Filter Type	Application Point	Restrictions

Postservice filter
 	
Configure at the `[edit firewall family (inet | inet6)]` hierarchy level by including the following statement:
 	
`service-filter` *`service-filter-name`*;
	Family `inet` or `inet6` on a logical interface
	
	Apply at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family (inet | inet6)]` hierarchy level by including the `post-service-filter` statement to apply a service filter as an input filter:
	
	`service {`
	`  input {`
	`  post-service-filter` *`filter-name`*;
	`  }`
	`}`
		A postservice filter is applied to traffic returning to the services interface after service processing. The filter is applied only if a service set is configured and selected.

Simple filter Configure at the `[edit firewall family inet]` hierarchy level by including the following statement: `simple-filter` *`filter-name`*	Family `inet` on a logical interface Apply at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet]` hierarchy level by including the following statement: `simple-filter` *`simple-filter-name`*;	Simple filters can only be applied as input filters. Supported on the following platforms only: • Gigabit Ethernet intelligent queuing (IQ2) PICs on the M120, M320, and T Series routers. • Enhanced Queuing Dense Port Concentrators (EQ DPC) on MX Series routers (and EX Series switches).

Table 28: Stateless Firewall Filter Configuration and Application Summary *(Continued)*

Filter Type	Application Point	Restrictions

Reverse packet forwarding (RPF) check filter
 	
Configured at the `[edit firewall family (inet | inet6)]` hierarchy level by including the following statement:
 	
`filter filter-name;`
	Family `inet` or `inet6` on a logical interface
	
	Apply at the `[edit interfaces interface-name unit unit-number family (inet | inet6)]` hierarchy level by including the following statement:
	
	`rpf-check fail-filter filter-name`
	
	to apply the stateless firewall filter as an RPF check filter.
	
	`rpf-check {`
	`  fail-filter filter-name;`
	`  mode loose;`
	`}`
		Supported on MX Series routers and EX Series switches only.

How Standard Firewall Filters Evaluate Packets

IN THIS SECTION

- Firewall Filter Packet Evaluation Overview | **794**
- Packet Evaluation at a Single Firewall Filter | **795**
- Best Practice: Explicitly Accept Any Traffic That Is Not Specifically Discarded | **796**
- Best Practice: Explicitly Reject Any Traffic That Is Not Specifically Accepted | **797**
- Multiple Firewall Filters Attached to a Single Interface | **797**
- Single Firewall Filter Attached to Multiple Interfaces | **797**

This topic covers the following information:

Firewall Filter Packet Evaluation Overview

The following sequence describes how the device evaluates a packet entering or exiting an interface if the input or output traffic at a device interface is associated with a *firewall filter*. Packet evaluation proceeds as follows:

1. The device evaluates the packet against the terms in the firewall filter sequentially, beginning with the first term in the filter.

   - If the packet matches all the conditions specified in a term, the device performs all the actions specified in that term.
   - If the packet does not match all the conditions specified in a term, the device proceeds to the next term in the filter (if a subsequent term exists) and evaluates the packet against that term.
   - If the packet does not match any term in the firewall filter, the device implicitly discards the packet.

2. Unlike service filters and simple filters, firewall filters support the `next term` action, which is neither a terminating action nor a nonterminating action but a flow control action.

   NOTE: On Junos and Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

   - If the matched term includes the `next term` action, the device continues evaluation of the packet at the next term within the firewall filter.
   - If the matched term does not include the `next term` action, evaluation of the packet against the given firewall filter ends at this term. The device does not evaluate the packet against any subsequent terms in this filter.

   A maximum of 1024 `next term` actions are supported per firewall filter configuration. If you configure a firewall filter that exceeds this limit, your candidate configuration results in a commit error.

3. The device stops evaluating a packet against a given firewall filter when either the packet matches a term without the `next term` action or the packet fails to match the last term in the firewall filter.

4. If a local packet arrives at a router interface that is associated with an ingress firewall filter, the filter evaluates the packet twice. The first evaluation occurs in the Packet Forwarding Engine, which is the central processing element of the router's forwarding plane, and the second evaluation occurs in the Routing Engine, which runs the router's control plane software.

NOTE: Local packets--chunks of data that are destined for or sent by the router itself--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).

If the first evaluation of the firewall filter modifies the incoming local packet or packet context values, the second evaluation of the firewall filter is based on the updated packet or packet context values.

For example, suppose that the filter includes a match condition based on the forwarding class or loss priority value associated with the packet and that the filter includes an action that modifies the forwarding class or loss priority value associated with the packet. If an ingress local packet arrives at an associated interface and the filter evaluation in the Packet Forwarding Engine modifies (rather than drops) the packet, then the filter evaluation in the Routing Engine is based on the modified packet context (rather than the original packet context).

Packet Evaluation at a Single Firewall Filter

Table 29 on page 795 describes packet-filtering behaviors at a device interface associated with a single firewall filter.

NOTE: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

Table 29: Packet Evaluation at a Single Firewall Filter

Firewall Filter Event	Action	Subsequent Action
The firewall filter term does not specify any match conditions.	The term matches all packets by default, and so the device performs the actions specified by that term.	If the term actions include the `next term` action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).

Table 29: Packet Evaluation at a Single Firewall Filter *(Continued)*

Firewall Filter Event	Action	Subsequent Action
The packet matches all conditions specified by the firewall filter term.	The device performs the actions specified by that term.	If the term actions include the `next term` action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).
The packet matches all conditions specified by the firewall filter term, but the term does not specify any actions.	The device implicitly accepts the packet.	If the term actions include the `next term` action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).
The packet does not match all conditions specified by the firewall filter term.	The device does not perform the actions specified by that term.	The device continues evaluation of the packet against the next term within the filter (if a subsequent term exists).

The packet does not match any term in the filter	The device implicitly discards the packet
	
	Every firewall filter configuration includes an implicit `discard` action at the end of the filter. This implicit terminating action is equivalent to including the following example term `t_explicit_discard` as the final term in the firewall filter:
	
	`term t_explicit_discard {`
	`  then discard;`
	`}`	

Best Practice: Explicitly Accept Any Traffic That Is Not Specifically Discarded

You might want a firewall filter to accept any traffic that the filter does not specifically discard. In this case, we recommend that you configure the firewall filter with a final term that specifies the `accept` terminating action.

In the following example snippet, configuring the `t_allow_all_else` term as the final term in the firewall filter explicitly configures the firewall filter to accept any traffic that the filter did not specifically discard :

```
term t_allow_all_else {
    then accept;
}
```

Following this best practice can simplify troubleshooting of the firewall filter.

Best Practice: Explicitly Reject Any Traffic That Is Not Specifically Accepted

On the other hand, you might want a firewall filter to reject any traffic that the firewall filter does not specifically accept. In this case, we recommend that you configure the firewall filter with a final term that specifies the `reject` terminating action.

In the following example snippet, configuring the `t_deny_all_else` term as the final term in the firewall filter explicitly configures the firewall filter to reject any traffic that the filter did not specifically accept:

```
term t_deny_all_else {
    then reject;
}
```

Following this best practice can simplify troubleshooting of the firewall filter.

Multiple Firewall Filters Attached to a Single Interface

On supported device interfaces, you can attach multiple firewall filters to a single interface. For more information, see "Understanding Multiple Firewall Filters Applied as a List" on page 1338.

NOTE: On supported interfaces, you can attach a protocol-independent (`family any`) firewall filter and a protocol-specific (`family inet` or `family inet6`) firewall filter to the same interface. The protocol-independent firewall filter executes first. For more information, see "Guidelines for Applying Standard Firewall Filters" on page 821.

Single Firewall Filter Attached to Multiple Interfaces

On supported interfaces, you can associate a single firewall filter with multiple interfaces, and Junos OS creates an *interface-specific instance* of that firewall filter for each associated interface.

- Junos OS associates each interface-specific instantiation of a firewall filter with a system-generated, interface-specific name.
- For any `count` actions in the filter terms, the Packet Forwarding Engine maintains separate, interface-specific counters, and Junos OS associates each counter with a system-generated, interface-specific name.
- For any `policer` actions in the filter terms, Junos OS creates separate, interface-specific instances of the policer actions.

For more information, see "Interface-Specific Firewall Filter Instances Overview" on page 1371.

RELATED DOCUMENTATION

Firewall Filter Match Conditions for Protocol-Independent Traffic | 937
Firewall Filter Terminating Actions | 884
How Service Filters Evaluate Packets | 1433
How Simple Filters Evaluate Packets | 1464
Guidelines for Configuring Firewall Filters | 814
Understanding How to Use Standard Firewall Filters | 778

Understanding Firewall Filter Fast Lookup Filter

In order to enhance the speed at which specific firewall filters are processed, you can use the filter block hardware available on certain modular port concentrators (MPCs). See the MX Series Interface Module Reference manual for details.This hardware allows for an increase in the number of firewall filter operations per second that can be accomplished.

Using the `fast-lookup-filter` option in environments with hundreds or thousands of terms per filter can increase performance of those filters by utilizing the filter block hardware.

There are 4096 hardware filters available per MPC. The number of firewall filters that can be installed in hardware depends on the number of terms in each filter. One hardware filter is needed for every group of 255 terms in a firewall filter. The total number of terms supported per firewall filter is 8000. However, attaching a given firewall filter with less than 256 terms to multiple interfaces will only result in one instance of that firewall filter being installed in the filter block. This is true for interface-specific filters as well as for filter lists.

You designate specific firewall filters to be processed in the filter block hardware by including the `fast-lookup-filter` option when configuring the firewall.

When this option is used, firewall parameters are stored in the filter block hardware which accelerates the lookup process. `fast-lookup-filter` is only available for the inet and inet6 protocol families. The match conditions are limited to 5-tuples: `protocol`, `source-address`, `destination-address`, `source-port`, and `destination-port`.

Ranges, prefix lists, and the except keyword are supported within the firewall filters and terms when using this option.

NOTE: Firewall filters that are configured using the `fast-lookup-filter` option are not optimized by the firewall compiler.

RELATED DOCUMENTATION

MX Series 5G Universal Routing Platform Interface Module Reference

fast-lookup-filter

Understanding Egress Firewall Filters with PVLANs

If you apply firewall filters to private VLANs in the output direction, the behavior of the filters might be unexpected. This topic explains how egress filters behave when applied to private VLANs.

If you apply a *firewall filter* in the output direction to a primary VLAN, the filter also applies to the secondary VLANs that are members of the primary VLAN when the traffic egresses with the primary VLAN tag or isolated VLAN tag, as listed below:

- Traffic forwarded from a secondary VLAN trunk port to a promiscuous port (trunk or access)
- Traffic forwarded from a secondary VLAN trunk port to a PVLAN trunk port.
- Traffic forwarded from a promiscuous port (trunk or access) to a secondary VLAN trunk port
- Traffic forwarded from a PVLAN trunk port. to a secondary VLAN trunk port
- Traffic forwarded from a community port to a promiscuous port (trunk or access)

If you apply a firewall filter in the output direction to a primary VLAN, the filter does *not* apply to traffic that egresses with a community VLAN tag, as listed below:

- Traffic forwarded from a community trunk port to a PVLAN trunk port
- Traffic forwarded from a promiscuous port (trunk or access) to a community trunk port

- Traffic forwarded from a PVLAN trunk port. to a community trunk port

If you apply a firewall filter in the output direction to a community VLAN, the following behaviors apply:

- The filter is applied to traffic forwarded from a promiscuous port (trunk or access) to a community trunk port (because the traffic egresses with the community VLAN tag).
- The filter is applied to traffic forwarded from a community port to a PVLAN trunk port (because the traffic egresses with the community VLAN tag).
- The filter is *not* applied to traffic forwarded from a community port to a promiscuous port (because the traffic egresses with the primary VLAN tag or untagged).

RELATED DOCUMENTATION

- Understanding Private VLANs
- Creating a Private VLAN on a Single QFX Switch
- Configuring VLANs on Switches
- Creating a Private VLAN Spanning Multiple QFX Series Switches

Selective Class-based Filtering on PTX Routers

SUMMARY

IN THIS SECTION

- Selective Class-based Filtering on PTX Routers | **800**
- Understanding Class-based Filtering on PTX Routers | **802**
- Example: Selective Class Based Filtering (PTX Routers) | **803**

Selective Class-based Filtering on PTX Routers

For supported PTX Series routers and line cards, you can filter IPv4 and IPv6 traffic based on the source or destination classification (Source Class Usage, SCU) and (Destination Class Usage, DCU). This is useful because it means you can apply a filter selectively, on a subset of packets in a class, rather than all

packets in the class. In addition, the packet flow through the packet forwarding engine (PFE) is optimized, and the filtering is more efficient.

For service providers, class-based filtering allows you to provide advanced services such as:

- Per-hop behavior manipulation by adjusting the forwarding class of the packet based on the source or destination packet class and other filter criteria.
- Traffic rate limiting towards certain customer interfaces, with high volume of traffic to drop (for example, under DDoS attack). Normally, you would deploy an outgoing interface filter to rate limit the traffic. However, this may be inefficient because traffic still crosses the fabric in distributed systems and consumes limited fabric bandwidth. The inefficiency becomes even more visible in a Virtual Output Queueing system, like PTX, where admission into the egress queue is happening before the output filter is executed and any subsequent drop action in the output filter requires compensation – more traffic needs to be admitted into the queue, which requires more fabric bandwidth and more egress on-chip buffer space (which is a limited resource). Class-based filters are executed in the ingress pipeline before admission of the packets into the egress queue. This mechanism is recommended over regular output interface filters, if you expect to drop large volumes of traffic towards certain destinations.

Class-based filtering is also effective for "low-and-slow" DoS attacks that target application and server resources by mimicking normal traffic patterns.

To support class-based filtering, two new bind points are introduced at the forwarding table for PTX routers: `source-class` and `destination-class`.

The CLI hierarchy is shown here, where **src-class-name** or **dest-class-name** is the name of the filter you defined in the corresponding policy.

```
routing-options forwarding-table source-class src-class-name family [inet | inet6] filter
<filter-name>
```

```
routing-options forwarding-table destination-class dest-class-name family [inet | inet6] filter
<filter-name>
```

You can also configure instance-specific filters across multiple SCU and DCU classes. By default, only one set of counters and policers is instantiated for a filter. In the instance-specific filter, separate set of counters and policers is created for each filter attach point.

```
firewall family [inet | inet6] filter <filter name> instance-specific
```

Understanding Class-based Filtering on PTX Routers

Initially, Source Class Usage (SCU) feature was introduced to provide statistics breakdown of traffic sent towards specific interface per originating prefix (identified by the source class). Destination Class Usage (DCU) was originally introduced to provide statistics breakdown of traffic received on an interface per destination prefix (identified by the destination class).

Both source or destination classes are assigned to the packet in the source or destination lookup process. Therefore, source and destination filter match conditions can be evaluated only if the filter is executed after the lookup.

Juniper routers support multiple filter bind points, those that may leverage the result of the source and destination classification are listed below, with usage guidelines:

- Output interface filter (set interfaces <interface name> family inet filter <output>. Supported on any PTX platform, but not recommended if it is expected to discard large volumes of traffic in a steady state (for example, when implementing a DDoS attack mitigation filter). Discarded DDoS attack traffic may not be compensated by other traffic not matching DDoS attack criteria due to limited fabric bandwidth and limited egress on-chip buffer space.
- Filter after forwarding table filter lookup (set forwarding-options family inet filter <filter-name> output). Supported on Express 2 (PE) and Express 3 (ZX) platforms. However, the filters are instantiated in the egress pipeline, therefore the discard behavior is similar to the regular output interface filter.
- Source or destination class specific bind point (set routing-options forwarding-table source-class src-class-name family [inet | inet6] filter <filter- name>). Supported on Express 2 (PE), Express 3 (ZX) and Express 4 (BT) platforms. This filter is instantiated in the ingress pipeline. This is the recommended option to discard large volumes of traffic. This option is also recommended if you need to override forwarding class and subsequently output queue assignment. In a Virtual Output Queueing system, queue is selected in the ingress pipeline and any override must happen in the ingress pipeline too.

NOTE:

- Note, these filter actions are not supported in the filter bound to the source or destination class specific bind point:
  - routing-instance
  - next-ip
  - next-interface

    - decapsulate
    - encapsulate
- Selective class-based filters cannot be applied on host-bound packets.
- Packets which fail uRPF lookups, but are restored by uRPF fail-filters are not subject to SCU/DCU lookups. Hence, selective class-based filters cannot be applied on such packets.
- Filters are applied only to packets ingressing on interfaces which have SCU/DCU feature enabled. This means filters would be applied irrespective of whether SCU is configured on output interfaces or not.
- Packets for which selective class-based filter needs to be applied may cause drop in performance. Performance drop would be function of rate of incoming traffic, average packets size, and amount of traffic subjected to the filters. However, packets on which selective class-based filters are not applied, do not affect performance.
- DCU accounting is applicable for packets dropped by filters.
- SCU output accounting is not applicable for packets dropped by filters.
- Selective class-based filters cannot be used with interface-specific knob because this knob is only applicable to interface-attached filters.
- Lists (input/output lists) of selective class-based filters are not supported.
- Logical systems are not supported.
- Only IPv4 and IPv6 are the supported payload protocols. MPLS is not supported.
- If a packet matches both SCU and DCU selective class-based filters then only the last filter (i.e., DCU filter) is applied to the packet and but not both filters.

Example: Selective Class Based Filtering (PTX Routers)

IN THIS SECTION

- Requirements | **804**
- Overview | **804**
- Configuration | **807**

This example shows how to apply firewall actions (discard, reject, or police) to IPv4 and IPv6 traffic flows on the basis of source or destination classification. It applies to PTX10001-36MR, PTX10003-160C, PTX10003-80C, PTX10004, and PTX10008 routers running Junos Evolved OS release 21.2, PTX10016 routers running JUNOS Evolved OS release 21.4, or PTX3000, PTX-5000, PTX1000, PTX10002, PTX10008, PTX10016 routers running Junos OS release 21.2 or later software.

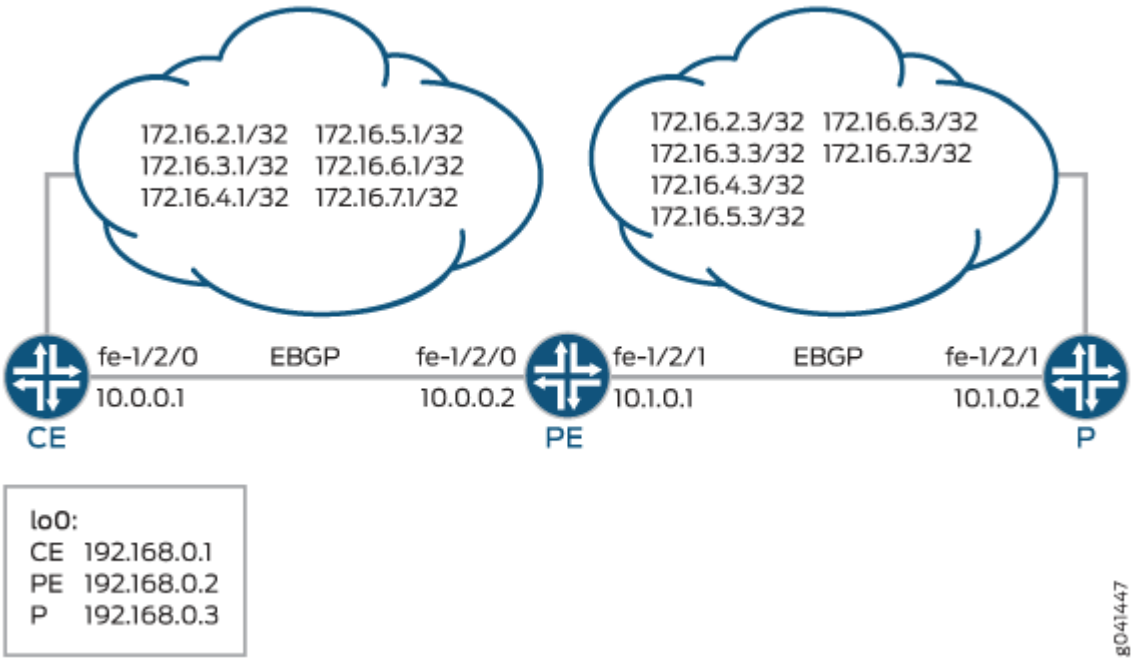
Requirements

This example uses BGP because BGP can be used to exchange routes between devices in a network topology consisting of customer edge, provider edge, and provider routers. Refer BGP Configuration Overview to read more.

Overview

This example uses three routing devices: a customer edge (CE) device, a provider edge (PE) device, and a provider core (P) device. The configuration for IPv4 traffic is shown, and includes two sets of SCU and DCU classes, plus the firewall filters. In the following figure, the /32 IP prefixes represent hosts connected to the customer edge (CE) and provider (P) routers respectively.

Figure 49: Sample Network



In this example, we define two classes of traffic: **scu-1** and **scu-2**, the first one is assigned to prefixes in the subnet 172.16.2.0/24 and the second one is assigned to prefixes in the subnet 172.16.3.0/24.

Other prefixes do not have any class assignments. As shown in the following CLI snippet, routing policy is defined on the PE router to assign prefixes to source-class scu-1 and source-class scu-2.

```
show policy-options
policy-statement scu-class {
    term gold {
        from {
            route-filter 172.16.2.0/24 orlonger;
              }
        then source-class scu-1;
        accept;
    term silver {
        from {
            route-filter 172.16.3.0/24 orlonger;
              }
        then source-class scu-2;
        accept;
        }
```

To account for traffic ingressing PE's interfaces from CE, the policy called **dcu-class** defined on the PE router uses route filters to place traffic into **dcu-1**, other prefixes have no class assignments.

```
show policy-options
policy-statement dcu-class {
    term gold {
        from {
            route-filter 172.16.5.0/24 orlonger;
            }
        then destination-class dcu-1;
        accept;
        }
```

The policies are then applied to the forwarding table.

```
forwarding-table {
    export [ dcu_class scu_class ];
}
```

In the next step we configure a filter on the PE router.

```
show firewall {
    family inet {
        filter f1 {
            term t1 {
                from {
                    protocol icmp;
                    }
                then {
                    count c1;
                    }
                }
            }
        }
    }
```

And attach that filter to the specific source and destination class bind points on the PE router.

```
show routing-options forwarding-table
source-class scu-1 {
    family inet {
        filter {
            f1;
        }
    }
}
destination-class dcu-1 {
    family inet {
        filter {
            f1;
        }
    }
}
```

Configuration

Procedure

CLI Quick Configuration

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

The example uses static routes to provide connectivity and loopback interface addresses for testing the operation.

Device CE

```
set interfaces et-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set policy-options policy-statement send-direct term 1 from protocol direct set policy-options
policy-statement send-direct term 1 then accept
set policy-options policy-statement send-static term 1 from protocol static set policy-options
policy-statement send-static term 1 then accept
set routing-options static route 10.1.0.0/30 next-hop 10.0.0.2 set routing-options autonomous-
system 100
```

Device PE

```
set interfaces et-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces et-1/2/1 unit 0 family inet accounting source-class-usage input
set interfaces et-1/2/1 unit 0 family inet accounting destination-class-usage
set interfaces et-1/2/1 unit 0 family inet address 10.1.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
```

```
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
set policy-options policy-statement dcu_class term gold from route-filter 172.16.5.0/24 orlonger
set policy-options policy-statement dcu_class term gold then destination-class dcu-1
set policy-options policy-statement scu_class term gold from route-filter 172.16.2.0/24 orlonger
set policy-options policy-statement scu_class term gold then source-class scu-1
set policy-options policy-statement scu_class term silver from route-filter 172.16.3.0/24
orlonger
set policy-options policy-statement scu_class term silver then source-class scu-1
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set firewall family inet filter f1 term 0 from protocol icmp
set firewall family inet filter f1 term 0 then count c1
set firewall family inet filter f1 term 0 then accept
set routing-options autonomous-system 200
set routing-options forwarding-table export dcu_class
set routing-options forwarding-table export scu_class
```

Device P

```
set interfaces et-1/2/1 unit 0 family inet address 10.1.0.2/30
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set interfaces lo0 unit 0 family inet address 172.16.0.3/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext export send-static
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.1.0.1
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set policy-options policy-statement send-static term 1 from protocol static
set policy-options policy-statement send-static term 1 then accept
set routing-options static route 10.0.0.0/30 next-hop 10.1.0.1
set routing-options static route 172.16.2.0/24 discard
set routing-options static route 172.16.3.0/24 discard
set routing-options static route 172.16.4.0/24 discard set routing-options static route
```

```
172.16.5.0/24 discard set routing-options static route 172.16.6.0/24 discard set routing-options
static route 172.16.7.0/24 discard set routing-options autonomous-system 300
```

Step-by-Step Procedure

To group source and destination prefixes in a forwarding class:

1. Create the router interfaces on the PE router.

```
[edit interfaces]
set et-1/2/0 unit 0 family inet accounting source-class-usage output
set et-1/2/0 unit 0 family inet address 10.0.0.2/30
set et-1/2/1 unit 0 family inet accounting source-class-usage input
set et-1/2/1 unit 0 family inet accounting destination-class-usage
set et-1/2/1 unit 0 family inet address 10.1.0.1/30
set lo0 unit 0 family inet address 192.168.0.2/32
```

2. Configure BGP on PE router.

```
[edit]
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols bgp group ext neighbor 10.1.0.2 peer-as 300
```

3. Configure the autonomous system (AS) number of the PE router.

```
[edit]
set routing-options autonomous-system 200
```

4. Configure the DCU policy on the PE router.

```
[edit]
set policy-options policy-statement dcu_class term class-1 from route-filter 172.16.5.0/24
```

```
orlonger
set policy-options policy-statement dcu_class term class-1 then destination-class dcu-1
```

5. Configure the SCU policy on the PE router.

```
[edit]
set policy-options policy-statement scu_class term class-1 from route-filter 172.16.2.0/24
orlonger
set policy-options policy-statement scu_class term class-1 then source-class scu-1
```

6. Apply the policies to the forwarding table on the PE router.

```
[edit]
set routing-options forwarding-table export dcu_class
set routing-options forwarding-table export scu_class
```

7. Create the filter on the PE router.

```
[edit]
set firewall family inet filter f1 from protocol icmp then count c1
```

8. Bind the filter to the source class and destination class bind points on the PE router.

   Binding the filter to destination class usage.

```
[edit]
set routing-options forwarding-table destination-class dcu-1 family inet filter f1
```

   Binding the filter to source class usage.

```
[edit]
set routing-options forwarding-table source-class scu-1 family inet filter f1
```

9. (Optional) Configure a routing policy that advertises direct routes on the PE router.

```
[edit]
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
```

Results

From configuration mode , confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands on the PE router. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
show interfaces
et-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
            }
        }
     }
et-1/2/1 {
    unit 0 {
        family inet {
            accounting {
                source-class-usage { input;
                    }
            }
            address 10.1.0.1/30;
            }
        }
     }
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.2/32;
                }
```

```
            }
        }
```

```
show interface statistics et-1/2/0
Physical interface: et-1/2/0:0, Enabled, Physical link is Up
    Interface index: 1087, SNMP ifIndex: 622
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Speed: 10Gbps, BPDU Error: None, Loop
Detect PDU Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Media type: Fiber Device flags    :
Present Running
    Interface flags: SNMP-Traps
    CoS queues    : 8 supported, 8 maximum usable queues
    Current address: e4:5d:37:4e:e8:40, Hardware address: e4:5d:37:4e:e8:40 Last flapped    :
2021-03-16 09:33:43 PDT (03:39:51 ago)
    Statistics last cleared: 2021-03-16 13:13:01 PDT (00:00:33 ago) Input rate    : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps) Input errors: 0, Output errors: 0 Active alarms : None
    Active defects : None
    PCS statistics    Seconds
        Bit errors    0
        Errored blocks    0
    PRBS Mode : Disabled
    Interface transmit statistics: Disabled Link Degrade :
        Link Monitoring    :  Disable

    Logical interface et-1/2/0:0.0 (Index 1047) (SNMP ifIndex 673)
        Flags: Up SNMP-Traps
        Encapsulation: ENET2
        Input packets : 14
        Output packets: 6
        Protocol inet, MTU: 1500
        Flags: Sendbcast-pkt-to-re, SCU-out
                            Packets     Bytes
       Source class (packet-per-second) (bits-per-second)
                scu-1  6   504
                   (   0)(    0)
    Addresses, Flags: Is-Preferred Is-Primary
        Destination: 44.4.4/24, Local: 44.4.4.4, Broadcast: 44.4.4.255
    Protocol inet6, MTU: 1500
        Flags: None, SCU-out
                    Packets  Bytes
```

```
        Source class (packet-per-second) (bits-per-second)
            scu-1    0    0
            (        0) ( 0)
        Addresses, Flags: Is-Preferred Is-Primary
            Destination: 4001::/64, Local: 4001::4001
        Addresses, Flags: Is-Preferred
            Destination: fe80::/64, Local: fe80::e65d:37ff:fe4e:e840
        Protocol multiservice, MTU: Unlimited
            Flags: None
```

```
show firewall
Filter: f1
Counters:
Name        Bytes        Packets
c1            0            0
Filter: v4_instance_new-scu-scu-1
Counters:
Name                           Bytes        Packets
c_v4_instance_new-scu-scu-1       504            6
Filter: v6_instance_new-scu-scu-1
Counters:
Name                           Bytes        Packets
c_v6_instance_new-scu-scu-1        0            0
```

```
show policy-options
policy-statement dcu_class {
    term class-1 {
        from {
            route-filter 172.16.5.0/24 orlonger;
            }
        then destination-class dcu-1;
            }
        }
policy-statement scu_class {
    term class-1 {
        from {
            route-filter 172.16.2.0/24 orlonger;
            }
```

```
        then source-class scu-1;
        }
       }
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
show routing-options
autonomous-system 200;
forwarding-table {
    export [ dcu_class scu_class ];
}
```

If you are done configuring the device, enter commit from configuration mode.

Guidelines for Configuring Firewall Filters

IN THIS SECTION

- Statement Hierarchy for Configuring Firewall Filters | **815**
- Firewall Filter Protocol Families | **816**
- Firewall Filter Names and Options | **817**
- Firewall Filter Terms | **817**
- Firewall Filter Match Conditions | **818**
- Firewall Filter Actions | **820**

This topic covers the following information:

Statement Hierarchy for Configuring Firewall Filters

To configure a standard firewall filter, you can include the following statements. For an IPv4 standard firewall filter, the `family inet` statement is optional. For an IPv6 standard firewall filter, the `family inet6` statement is mandatory.

```
firewall {
    family family-name {
        filter filter-name {
            accounting-profile name;
            instance-shared;
            interface-specific;
            physical-interface-filter;
            term term-name {
                filter filter-name;
            }
            term term-name {
                from {
                    match-conditions;
                    ip-version ip-version {
                        match-conditions;
                        protocol (tcp | udp) {
                            match conditions;
                        }
                    }
                }
                then {
                    actions;
                }
            }
        }
    }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- `[edit]`
- `[edit logical-systems logical-system-name]`

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 toward the tunnel destination. The firewall filter affects only the packets exiting the router (or switch) by way of the tunnel.

NOTE: On ACX7100 platforms, VPLS firewall filters are configured under `family ethernet-switching` and not under `family VPLS`. Management filters are configured at family `inet` or `inet6` and the syntax is of this form:

```
set interfaces re0:mgmt-0 unit logical-unit-number family family-name filter input filter-name.
```

Firewall Filter Protocol Families

A firewall filter configuration is specific to a particular protocol family. Under the `firewall` statement, include one of the following statements to specify the protocol family for which you want to filter traffic:

- `family any`—To filter protocol-independent traffic.
- `family inet`—To filter Internet Protocol version 4 (IPv4) traffic.
- `family inet6`—To filter Internet Protocol version 6 (IPv6) traffic.
- `family mpls`—To filter MPLS traffic.
- `family vpls`—To filter virtual private LAN service (VPLS) traffic.
- `family ccc`—To filter Layer 2 circuit cross-connection (CCC) traffic.
- `family bridge`—To filter Layer 2 bridging traffic for MX Series 3D Universal Edge Routers only.
- `family ethernet-switching`—To filter Layer 2 (Ethernet) traffic.

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

NOTE: For bridge family filter, the *ip-protocol* match criteria is supported only for IPv4 and not for IPv6. This is applicable for line cards that support the Junos Trio chipset such as the MX 3D MPC line cards.

Firewall Filter Names and Options

Under the `family` *family-name* statement, you can include `filter` *filter-name* statements to create and name firewall filters. 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 (“ ”).

At the `[edit firewall family` *family-name* `filter` *filter-name*`]` hierarchy level, the following statements are optional:

- `accounting-profile`
- `instance-shared` (MX Series routers with Modular Port Concentrators (MPCS) only)
- `interface-specific`
- `physical-interface-filter`

Firewall Filter Terms

Under the `filter` *filter-name* statement, you can include `term` *term-name* statements to create and name filter terms.

- You must configure at least one term in a firewall filter.
- You must specify a unique name for each term within a firewall filter. The term name 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 (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter 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` configuration mode command to reorder the terms of a firewall filter.

At the `[edit firewall family` *family-name* `filter` *filter-name* `term` *term-name*`]` hierarchy level, the `filter` *filter-name* statement is not valid in the same term as `from` or `then` statements. When included at this hierarchy level, the `filter` *filter-name* statement is used to *nest* firewall filters.

Firewall Filter Match Conditions

Firewall filter match conditions are specific to the type of traffic being filtered.

With the exception of MPLS-tagged IPv4 or IPv6 traffic, you specify the term's match conditions under the `from` statement. For MPLS-tagged IPv4 traffic, you specify the term's IPv4 address-specific match conditions under the `ip-version ipv4` statement and the term's IPv4 port-specific match conditions under the `protocol (tcp | udp)` statement.

For MPLS-tagged IPv6 traffic, you specify the term's IPv6 address-specific match conditions under the `ip-version ipv6` statement and the term's IPv6 port-specific match conditions under the `protocol (tcp | udp)` statement.

Table 30 on page 818 describes the types of traffic for which you can configure firewall filters.

Table 30: Firewall Filter Match Conditions by Protocol Family

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
Protocol-independent	`[edit firewall family any filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Firewall Filter Match Conditions for Protocol-Independent Traffic" on page 937.
IPv4	`[edit firewall family inet filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Firewall Filter Match Conditions for IPv4 Traffic" on page 940.
IPv6	`[edit firewall family inet6 filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see " Firewall Filter Match Conditions for IPv6 Traffic" on page 957.
MPLS	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS Traffic" on page 1000.
IPv4 addresses in MPLS flows	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `ip-version ipv4 ]` For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 1009.

Table 30: Firewall Filter Match Conditions by Protocol Family *(Continued)*

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
IPv4 ports in MPLS flows	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `ip-version ipv4 protocol (tcp | udp)]` For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 1009.
IPv6 addresses in MPLS flows	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `ip-version ipv6 ]` For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 1009.
IPv6 ports in MPLS flows	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `ip-version ipv6 protocol (tcp | udp)]` For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 1009.
VPLS	`[edit firewall family vpls filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Firewall Filter Match Conditions for VPLS Traffic" on page 1013.
Layer 2 CCC	`[edit firewall family ccc filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Firewall Filter Match Conditions for Layer 2 CCC Traffic" on page 1032.
Layer 2 Bridging (MX Series routers and EX Series switches only)	`[edit firewall family bridge filter` *`filter-name`* `term` *`term-name`*`]` `[edit firewall family ethernet-switching filter` *`filter-name`* `term` *`term-name`*`]` (for EX Series switches only) For the complete list of match conditions, see "Firewall Filter Match Conditions for Layer 2 Bridging Traffic" on page 1037.

If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see IPv6 Overview and Supported IPv6 Standards.

Firewall Filter Actions

Under the `then` statement for a firewall filter term, you can specify the actions to be taken on a packet that matches the term.

Table 31 on page 820 summarizes the types of actions you can specify in a firewall filter term.

Table 31: Firewall Filter Action Categories

Type of Action	Description	Comment
Terminating	Halts all evaluation of a firewall filter for a specific packet. The router (or switch) performs the specified action, and no additional terms are used to examine the packet. You can specify only one *terminating action* in a firewall filter term. If you try to specify more than one *terminating action* within the filter term then the latest *terminating action* will replace the existing *terminating action*. You can, however, specify one terminating action with one or more *nonterminating actions* in a single term. For example, within a term, you can specify `accept` with `count` and `syslog`. Regardless of the number of terms that contain terminating actions, once the system processes a terminating action within a term, processing of the entire firewall filter halts.	See "Firewall Filter Terminating Actions" on page 884.
Nonterminating	Performs 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), but any additional terms are used to examine the packet.	All nonterminating actions include an implicit accept action. This accept action is carried out if no other terminating action is configured in the same term. See "Firewall Filter Nonterminating Actions" on page 871.

Table 31: Firewall Filter Action Categories *(Continued)*

Type of Action	Description	Comment
Flow control	For standard firewall filters only, the `next term` action directs the router (or switch) to perform configured actions on the packet and then, rather than terminate the filter, use the next term in the filter to evaluate the packet. If the `next term` action is included, the matching packet is 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 nonterminating action `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.	You cannot configure the `next term` action with a terminating action in the same filter term. However, you can configure the next term action with another nonterminating action in the same filter term. A maximum of 1024 `next term` actions are supported per standard firewall filter configuration. If you configure a standard firewall filter that exceeds this limit, your candidate configuration results in a commit error. **NOTE**: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

RELATED DOCUMENTATION

Guidelines for Applying Standard Firewall Filters | 821

Understanding How to Use Standard Firewall Filters | 778

Guidelines for Applying Standard Firewall Filters

IN THIS SECTION

- Applying Firewall Filters Overview | 822
- Statement Hierarchy for Applying Firewall Filters | 823
- Restrictions on Applying Firewall Filters | 825

Applying Firewall Filters Overview

You can apply a standard firewall filter to a loopback interface on the router or to a physical or logical interface on the router. You can apply a firewall filter to a single interface or to multiple interfaces on the router.Table 32 on page 822 summarizes the behavior of firewall filters based on the point to which you attach the filter.

Table 32: Firewall Filter Behavior by Filter Attachment Point

Filter Attachment Point	Filter Behavior
Loopback interface	The router's loopback interface, `lo0`, is the interface to the Routing Engine and carries no data packets. When you apply a firewall filter to the loopback interface, the filter evaluates the local packets received or transmitted by the Routing Engine. **NOTE**: • ACX5048 and ACX5096 routers do not support the evaluation of packets transmitted by the Routing engine for loopback interface filter.
Physical interface or logical interface	When you apply a filter to a physical interface on the router or to a logical interface (or member of an aggregated Ethernet bundle defined on the interface), the filter evaluates all data packet that pass through that interface.
Multiple interfaces	You can use the same firewall filter one or more times. On M Series routers, except the M120 and M320 routers, if you apply a firewall filter to multiple interfaces, the filter acts on the sum of traffic entering or exiting those interfaces. On T Series, M120, M320, and MX Series routers, interfaces are distributed among multiple packet-forwarding components. On these routers, you can configure firewall filters and service filters that, when applied to multiple interfaces, act on the individual traffic streams entering or exiting each interface, regardless of the sum of traffic on the multiple interfaces. For more information, see "Interface-Specific Firewall Filter Instances Overview" on page 1371.

Table 32: Firewall Filter Behavior by Filter Attachment Point *(Continued)*

Filter Attachment Point	Filter Behavior
Single interface with protocol-independent and protocol-specific firewall filters attached	For interfaces hosted on the following hardware only, you can attach a protocol-independent (`family any`) firewall filter and a protocol-specific (`family inet` or `family inet6`) firewall filter simultaneously. The protocol-independent firewall executes first. • ACX Series Universal Metro Routers • Flexible PIC Concentrators (FPCs) in M7i and M10i Multiservice Edge Routers • Modular Interface Cards (MICs) and Modular Port Concentrators (MPCs) in MX Series 5G Universal Routing Platforms • T Series Core Routers **NOTE**: Interfaces hosted on the following hardware do not support protocol-independent firewall filters: • Forwarding Engine Boards (FEBs) in M120 routers • Enhanced III FPCs in M320 routers • FPC2 and FPC3 modules in MX Series routers • Dense Port Concentrators (DPCs) in MX Series routers • PTX Series Packet Transport Routers

Statement Hierarchy for Applying Firewall Filters

To apply a standard firewall filter to a logical interface, configure the `filter` statement for the logical interface defined under either the `[edit]` or `[edit logical-systems` *`logical-system-name`*`]` hierarchy level. Under the `filter` statement, you can include one or more of the following statements: `group` *`group-number`*, `input` *`filter-name`*, `input-list` *`filter-name`*, `output` *`filter-name`*, or `output-list` *`filter-name`*. The hierarchy level at which you attach the `filter` statement depends on the filter type and device type you are configuring.

Protocol-Independent Firewall Filters on MX Series Routers

To apply a protocol-independent firewall filter to a logical interface on an MX Series router, configure the `filter` statement *directly* under the logical unit:

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

All Other Firewall Filters on Logical Interfaces

To apply a standard firewall filter to a logical interface for all cases *other than* a protocol-independent filter on an MX Series router, configure the `filter` statement under the protocol family:

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

Restrictions on Applying Firewall Filters

Number of Input and Output Filters Per Logical Interface

Input filters—Although you can use the same filter multiple times, you can apply only one input filter or one input filter list to an interface.

- To specify a single firewall filter to be used to evaluate packets received on the interface, include the `input` *`filter-name`* statement in the `filter` stanza.
- To specify an ordered list of firewall filters to be used to evaluate packets received on the interface, include the `input-list [` *`filter-names`* `]` statement in the `filter` stanza. You can specify up to 16 firewall filters for the filter input list.

Output filters—Although you can use the same filter multiple times, you can apply only one output filter or one output filter list to an interface.

- To specify a single firewall filter to be used to evaluate packets transmitted on the interface, include the `output` *`filter-name`* statement in the `filter` stanza.
- To specify an ordered list of firewall filters to be used to evaluate packets transmitted on the interface, include the `output-list [` *`filter-names`* `]` statement in the `filter` stanza. You can specify up to 16 firewall filters in a filter output list.

MPLS and Layer 2 CCC Firewall Filters in Lists

The `input-list` *`filter-names`* and `output-list` *`filter-names`* statements for firewall filters for the `ccc` and `mpls` protocol families are supported on all interfaces with the exception of the following:

- Management interfaces and internal Ethernet interfaces (`fxp` or `em0`)
- Loopback interfaces (`lo0`)
- USB modem interfaces (`umd`)

Layer 2 CCC Firewall Filters on MX Series Routers and EX Series Switches

Only on MX Series routers and EX Series switches, you cannot apply a Layer 2 CCC stateless firewall filter (a firewall filter configured at the `[edit firewall filter family ccc]` hierarchy level) as an output filter. On MX Series routers and EX Series switches, firewall filters configured for the `family ccc` statement can be applied only as input filters.

IPv6 Firewall Filters on PTX Series Packet Transport Routers

On PTX10001-20C routers, you cannot apply IPv6 firewall filters to:

- Tunnel interfaces
- IRB interfaces
- Egress interfaces
- Interface-specific filters, configured at the `[edit firewall family inet6 filter filter-name]` hierarchy level.
- Traffic policers
- Junos Telemetry Interface

RELATED DOCUMENTATION

family (Firewall)

family (Interfaces)

filter (Applying to a Logical Interface)

filter (Configuring)

Guidelines for Configuring Firewall Filters | 814

Understanding How to Use Standard Firewall Filters | 778

Supported Standards for Filtering

The Junos OS 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*

NOTE: ACX Series routers do not support RFC 2460, RFC 4291, and RFC 4443 standards.

RELATED DOCUMENTATION

Firewall Filters Overview | 772
Service Filter Overview | 1431
Simple Filter Overview | 1463
Firewall Filters in Logical Systems Overview | 1213

Monitoring Firewall Filter Traffic

IN THIS SECTION

Monitoring Traffic for All Firewall Filters and Policers That Are Configured | 827
Monitoring Traffic for a Specific Firewall Filter | 828
Monitoring Traffic for a Specific Policer | 829

You can use operational mode commands to monitor firewall filter traffic.

Monitoring Traffic for All Firewall Filters and Policers That Are Configured

IN THIS SECTION

Purpose | 828
Action | 828
Meaning | 828

Purpose

Monitor the number of packets and bytes that matched the firewall filters and monitor the number of packets that exceeded policer rate limits:

Action

Use the `show firewall` operational mode command:

```
user@switch> show firewall
Filter: egress-vlan-watch-employee
Counters:
Name                                                Bytes              Packets
counter-employee-web                                 3348                   27
Filter: ingress-port-limit-tcp-icmp
Counters:
Name                                                Bytes              Packets
icmp-counter                                             560                   10
Policers:
Name                                               Packets
icmp-connection-policer                                 10
tcp-connection-policer                                   0
Filter: ingress-vlan-rogue-block
Filter: ingress-vlan-limit-guest
```

Meaning

The `show firewall` command displays the names of all firewall filters, counters, and policers that are configured. For each counter that is specified in a filter configuration, the output field shows the byte count and packet count for the term in which the counter is specified. For each policer that is specified in a filter configuration, the output field shows the packet count for packets that exceed the specified rate limits.

Monitoring Traffic for a Specific Firewall Filter

IN THIS SECTION

Purpose | 829

- Action | 829
- Meaning | 829

Purpose

Monitor the number of packets and bytes that matched a firewall filter and monitor the number of packets that exceeded policer rate limits.

Action

Use the `show firewall filter` *`filter-name`* operational mode command:

```
user@switch> show firewall filter ingress-port-limit-tcp-icmp
Filter: ingress-port-limit-tcp-icmp
Counters:
Name                                                Bytes              Packets
icmp-counter                                          560                   10
```

Meaning

The `show firewall filter` *`filter-name`* command limits the display information to the counters and policers that are defined for the specified filter.

Monitoring Traffic for a Specific Policer

IN THIS SECTION

- Purpose | 829
- Action | 830
- Meaning | 830

Purpose

Monitor the number of packets that exceeded the rate limits of a policer:

Action

Use the `show firewall policer` *policer-name* operational mode command:

```
user@switch> show firewall policer icmp-connection-policer
Filter: ingress-port-limit-tcp-icmp
Policers:
Name                                                Packets
icmp-connection-policer                                  10
```

Meaning

The `show firewall policer` *policer-name* command displays the number of packets that exceeded the rate limits for the specified policer.

RELATED DOCUMENTATION

- Configuring Firewall Filters | **1826**
- Configuring Two-Color and Three-Color Policers to Control Traffic Rates | **2237**

Troubleshooting Firewall Filters

IN THIS SECTION

- Troubleshooting QFX10000 Switches | **831**
- Troubleshooting Other Switches | **832**

Use the following information to troubleshoot your firewall filter configuration.

Troubleshooting QFX10000 Switches

IN THIS SECTION

- Do Not Combine Match Conditions for Different Layers | 831
- Layer 2 Packets Cannot be Discarded with Firewall Filters | 831
- Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces | 832

This section describes issues specific to QFX10000 switches:

Do Not Combine Match Conditions for Different Layers

On QFX10000 switches, do not combine match conditions for Layer 2 and any other layer in a `family ethernet-switching` filter. (For example, do not include conditions that match MAC addresses and IP addresses in the same filter.) If you do so, the filter will commit successfully but will not work. You will also see the following log message: `L2 filter` *`filter-name`* `doesn't support mixed L2 and L3/L4 match conditions. Please re-config.`

Layer 2 Packets Cannot be Discarded with Firewall Filters

IN THIS SECTION

- Problem | 831
- Solution | 832

Problem

Description

Layer 2 (L2) control packets such as Link Layer Discovery Protocol (LLDP) and bridge protocol data unit (BPDU) cannot be discarded with firewall filters.

Solution

Configure distributed denial-of-service (DDoS) protection on the L2 control packet and set the aggregate policer bandwidth and burst values to the minimum value of 1. For example,

```
[edit system ddos-protection protocols protocol name]
```

user@host# **set aggregate bandwidth 1**

```
[edit system ddos-protection protocols protocol name]
```

user@host# **set aggregate burst 1**

Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces

IN THIS SECTION

- Problem | **832**
- Solution | **832**

Problem

Description

On QFX10000 switches, the Protect-RE (loopback) firewall filter does not filter packets applied to EM0 interfaces including SNMP, Telnet, and other services.

Solution

This is expected behavior.

Troubleshooting Other Switches

IN THIS SECTION

- Firewall Filter Configuration Returns a No Space Available in TCAM Message | **833**

- Filter Counts Previously Dropped Packet | **836**
- Matching Packets Not Counted | **837**
- Counter Reset When Editing Filter | **838**
- Cannot Include loss-priority and policer Actions in Same Term | **838**
- Cannot Egress Filter Certain Traffic Originating on QFX Switch | **839**
- Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **840**
- Egress Firewall Filters with Private VLANs | **840**
- Egress Filtering of L2PT Traffic Not Supported | **842**
- Cannot Drop BGP Packets in Certain Circumstances | **842**
- Invalid Statistics for Policer | **843**
- Policers can Limit Egress Filters | **843**

This section describes issues specific to QFX switches other than QFX10000 switches. This information also applies to OCX1100 switches and EX4600 switches.

Firewall Filter Configuration Returns a No Space Available in TCAM Message

IN THIS SECTION

- Problem | **833**
- Solution | **834**

Problem

Description

When a firewall filter configuration exceeds the amount of available Ternary Content Addressable Memory (TCAM) space, the system returns the following `syslogd` message:

```
No space available in tcam.
Rules for filter filter-name will not be installed.
```

A switch returns this message during the commit operation if the firewall filter that has been applied to a port, VLAN, or Layer 3 interface exceeds the amount of space available in the TCAM table. The filter is not applied, but the commit operation for the firewall filter configuration is completed in the CLI module.

Solution

When a firewall filter configuration exceeds the amount of available TCAM table space, you must configure a new firewall filter with fewer filter terms so that the space requirements for the filter do not exceed the available space in the TCAM table.

You can perform either of the following procedures to correct the problem:

To delete the filter and its binding and apply the new smaller firewall filter to the same binding:

1. Delete the filter and its binding to ports, VLANs, or Layer 3 interfaces. For example:

```
[edit]
user@switch# delete firewall family ethernet-switching filter ingress-vlan-rogue-block
user@switch# delete vlans employee-vlan description "filter to block rogue devices on
employee-vlan"
user@switch# delete vlans employee-vlan filter input ingress-vlan-rogue-block
```

2. Commit the changes:

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

3. Configure a smaller filter with fewer terms that does not exceed the amount of available TCAM space. For example:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block ...
```

4. Apply (bind) the new firewall filter to a port, VLAN , or Layer 3 interface. For example:

```
[edit]
user@switch# set vlans employee-vlan description "filter to block rogue devices on employee-
vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

5. Commit the changes:

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

To apply a new firewall filter and overwrite the existing binding but not delete the original filter:

1. Configure a firewall filter with fewer terms than the original filter:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block...
```

2. Apply the firewall filter to the port, VLAN, or Layer 3 interfaces to overwrite the binding of the original filter—for example:

```
[edit]
user@switch# set vlans employee-vlan description "smaller filter to block rogue devices on
employee-vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

   Because you can apply no more than one firewall filter per VLAN per direction, the binding of the original firewall filter to the VLAN is overwritten with the new firewall filter `new-ingress-vlan-rogue-block`.

3. Commit the changes:

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

NOTE: The original filter is not deleted and is still available in the configuration.

Filter Counts Previously Dropped Packet

IN THIS SECTION

- Problem | 836
- Solution | 837

Problem

Description

If you configure two or more filters in the same direction for a physical interface and one of the filters includes a counter, the counter will be incorrect if the following circumstances apply:

- You configure the filter that is applied to packets first to discard certain packets. For example, imagine that you have a VLAN filter that accepts packets sent to 10.10.1.0/24 addresses and implicitly discards packets sent to any other addresses. You apply the filter to the `admin` VLAN in the output direction, and interface xe-0/0/1 is a member of that VLAN.
- You configure a subsequent filter to accept and count packets that are dropped by the first filter. In this example, you have a port filter that accepts and counts packets sent to 192.168.1.0/24 addresses that is also applied to xe-0/0/1 in the output direction.

The egress VLAN filter is applied first and correctly discards packets sent to 192.168.1.0/24 addresses. The egress port filter is applied next and counts the discarded packets as matched packets. The packets are not forwarded, but the counter displayed by the egress port filter is incorrect.

Remember that the order in which filters are applied depends on the direction in which they are applied, as indicated here:

Ingress filters:

1. Port (Layer 2) filter
2. VLAN filter
3. Router (Layer 3) filter

Egress filters:

1. Router (Layer 3) filter
2. VLAN filter

3. Port (Layer 2) filter

Solution

This is expected behavior.

Matching Packets Not Counted

IN THIS SECTION

- Problem | 837
- Solution | 837

Problem

Description

If you configure two egress filters with counters for a physical interface and a packet matches both of the filters, only one of the counters includes that packet.

For example:

- You configure an egress port filter with a counter for interface xe-0/0/1.
- You configure an egress VLAN filter with a counter for the `admin`VLAN, and interface xe-0/0/1 is a member of that VLAN.
- A packet matches both filters.

In this case, the packet is counted by only one of the counters even though it matched both filters.

Solution

This is expected behavior.

Counter Reset When Editing Filter

IN THIS SECTION

- Problem | 838
- Solution | 838

Problem

Description

If you edit a firewall filter term, the value of any counter associated with any term in the same filter is set to 0, including the implicit counter for any policer referenced by the filter. Consider the following examples:

- Assume that your filter has `term1`, `term2`, and `term3`, and each term has a counter that has already counted matching packets. If you edit any of the terms in any way, the counters for all the terms are reset to 0.
- Assume that your filter has `term1` and `term2`. Also assume that `term2` has a `policer` action modifier and the implicit counter of the policer has already counted 1000 matching packets. If you edit `term1` or `term2` in any way, the counter for the policer referenced by `term2` is reset to 0.

Solution

This is expected behavior.

Cannot Include loss-priority and policer Actions in Same Term

IN THIS SECTION

- Problem | 839
- Solution | 839

Problem

Description

You cannot include both of the following actions in the same firewall filter term in a QFX Series switch:

- `loss-priority`
- `policer`

If you do so, you see the following error message when you attempt to commit the configuration: “cannot support policer action if loss-priority is configured.”

Solution

This is expected behavior.

Cannot Egress Filter Certain Traffic Originating on QFX Switch

IN THIS SECTION

- Problem | 839
- Solution | 839

Problem

Description

On a QFX Series switch, you cannot filter certain traffic with a firewall filter applied in the output direction if the traffic originates on the QFX switch. This limitation applies to control traffic for protocols such as ICMP (ping), STP, LACP, and so on.

Solution

This is expected behavior.

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling

IN THIS SECTION

- Problem | 840
- Solution | 840

Problem

Description

If you create a firewall filter that includes a match condition of `dot1q-tag` or `dot1q-user-priority` and apply the filter on input to a trunk port that participates in a service VLAN, the match condition does not work if the Q-in-Q EtherType is not 0x8100. (When Q-in-Q tunneling is enabled, trunk interfaces are assumed to be part of the service provider or data center network and therefore participate in service VLANs.)

Solution

This is expected behavior. To set the Q-in-Q EtherType to 0x8100, enter the **set dot1q-tunneling ethertype 0x8100** statement at the `[edit ethernet-switching-options]` hierarchy level. You must also configure the other end of the link to use the same Ethertype.

Egress Firewall Filters with Private VLANs

IN THIS SECTION

- Problem | 841
- Solution | 841

Problem

Description

If you apply a firewall filter in the output direction to a primary VLAN, the filter also applies to the secondary VLANs that are members of the primary VLAN when the traffic egresses with the primary VLAN tag or isolated VLAN tag, as listed below:

- Traffic forwarded from a secondary VLAN trunk port to a promiscuous port (trunk or access)
- Traffic forwarded from a secondary VLAN trunk port that carries an isolated VLAN to a PVLAN trunk port.
- Traffic forwarded from a promiscuous port (trunk or access) to a secondary VLAN trunk port
- Traffic forwarded from a PVLAN trunk port. to a secondary VLAN trunk port
- Traffic forwarded from a community port to a promiscuous port (trunk or access)

If you apply a firewall filter in the output direction to a primary VLAN, the filter does *not* apply to traffic that egresses with a community VLAN tag, as listed below:

- Traffic forwarded from a community trunk port to a PVLAN trunk port
- Traffic forwarded from a secondary VLAN trunk port that carries a community VLAN to a PVLAN trunk port
- Traffic forwarded from a promiscuous port (trunk or access) to a community trunk port
- Traffic forwarded from a PVLAN trunk port. to a community trunk port

If you apply a firewall filter in the output direction to a community VLAN, the following behaviors apply:

- The filter is applied to traffic forwarded from a promiscuous port (trunk or access) to a community trunk port (because the traffic egresses with the community VLAN tag).
- The filter is applied to traffic forwarded from a community port to a PVLAN trunk port (because the traffic egresses with the community VLAN tag).
- The filter is *not* applied to traffic forwarded from a community port to a promiscuous port (because the traffic egresses with the primary VLAN tag or untagged).

Solution

These are expected behaviors. They occur only if you apply a firewall filter to a private VLAN in the output direction and do not occur if you apply a firewall filter to a private VLAN in the input direction.

Egress Filtering of L2PT Traffic Not Supported

IN THIS SECTION

- Problem | 842
- Solution | 842

Problem

Description

Egress filtering of L2PT traffic is not supported on the QFX3500 switch. That is, if you configure L2PT to tunnel a protocol on an interface, you cannot also use a firewall filter to filter traffic for that protocol on that interface in the output direction. If you commit a configuration for this purpose, the firewall filter is not applied to the L2PT-tunneled traffic.

Solution

This is expected behavior.

Cannot Drop BGP Packets in Certain Circumstances

IN THIS SECTION

- Problem | 842
- Solution | 843

Problem

Description

BGP packets with a time-to-live (TTL) value greater than 1 cannot be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface. BGP packets with TTL value of 1 or 0 can be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface.

Solution

This is expected behavior.

Invalid Statistics for Policer

IN THIS SECTION

- Problem | 843
- Solution | 843

Problem

Description

If you apply a single-rate two-color policer in more than 128 terms in a firewall filter, the output of the `show firewall` command displays incorrect data for the policer.

Solution

This is expected behavior.

Policers can Limit Egress Filters

IN THIS SECTION

- Problem | 843
- Solution | 844

Problem

Description

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry

TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

Solution

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

CHAPTER 14

Firewall Filter Match Conditions and Actions

IN THIS CHAPTER

- Overview of Firewall Filters (OCX Series) | **846**
- Understanding Firewall Filter Match Conditions | **848**
- Understanding Firewall Filter Planning | **853**
- Understanding How Firewall Filters Are Evaluated | **854**
- Understanding Firewall Filter Match Conditions | **856**
- Firewall Filter Flexible Match Conditions | **862**
- Firewall Filter Nonterminating Actions | **871**
- Firewall Filter Terminating Actions | **884**
- Firewall Filter Match Conditions and Actions (ACX Series Routers) | **909**
- Firewall Filter Match Conditions for Protocol-Independent Traffic | **937**
- Firewall Filter Match Conditions for IPv4 Traffic | **940**
- Firewall Filter Match Conditions for IPv6 Traffic | **957**
- Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**
- Firewall Filter Match Conditions Based on Bit-Field Values | **976**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**
- Understanding IP-Based Filtering and Selective Port Mirroring of MPLS Traffic | **994**
- Firewall Filter Match Conditions for MPLS Traffic | **1000**
- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic | **1009**
- Firewall Filter Match Conditions for VPLS Traffic | **1013**
- Firewall Filter Match Conditions for Layer 2 CCC Traffic | **1032**
- Firewall Filter Match Conditions for Layer 2 Bridging Traffic | **1037**
- Firewall Filter Support on Loopback Interface | **1055**

Overview of Firewall Filters (OCX Series)

IN THIS SECTION

- Where You Can Apply Filters | 846
- Firewall Filter Components | 847
- Firewall Filter Processing | 848

Firewall filters provide rules that define whether to accept or discard packets that are transiting an interface. If a packet is accepted, you can configure additional actions to perform on the packet, such as class-of-service (CoS) marking (grouping similar types of traffic together and treating each type of traffic as a class with its own level of service priority) and traffic policing (controlling the maximum rate of traffic sent or received). You configure firewall filters to determine whether to accept or discard a packet before it enters or exits a Layer 3 (routed) interface.

An *ingress firewall filter* is applied to packets that are entering an interface, and an *egress* firewall filter is applied to packets that are exiting an interface .

NOTE: Firewall filters are sometimes called *access control lists* (ACLs).

Where You Can Apply Filters

You can apply a router firewall filter in both ingress and egress directions on IPv4 or IPv6 Layer 3 (routed) interfaces and a loopback interface, which filters traffic sent to the switch itself or generated by the switch.

You apply a filter to a loopback interface in the input direction to protect the switch from unwanted traffic. You also might want to apply a filter to a loopback interface in the output direction so that you can set the forwarding class and DSCP bit value for packets that originate on the switch itself. This feature gives you very fine control over the classification of CPU generated packets. For example, you might want to assign different DSCP values and forwarding classes to traffic generated by different routing protocols so the traffic for those protocols can be treated in a differentiated manner by other devices.

NOTE: On QFX5220 switches, you can only apply a filter to a loopback interface in the ingress direction.

NOTE: If you apply ingress and egress filters to the same interface, the ingress filter is processed first.

To apply a firewall filter:

1. Configure the firewall filter.
2. Apply the firewall filter to a Layer 3 interface and specify the direction. If you specify the `input` direction, traffic is filtered on ingress. If you specify the `output` direction, traffic is filtered on egress.

NOTE: You can apply only one firewall filter to a Layer 3 interface for a given direction. For example, for a given `family inet` interface, you can apply one filter for input and one for output.

OCX switches support the maximum numbers of *firewall filter* terms per type of attachment point shown in Table 33 on page 847.

Table 33: Supported Firewall Filter Numbers

Filter Type	Maximum Number of Filters
Ingress	1536
Egress	1024

Firewall Filter Components

In a firewall filter, you first define the family address type (`inet` for IPv4 or `inet6` for IPv6) and then define one or more terms that specify the filtering criteria and the action to take if a match occurs.

Each term consists of the following components:

- Match conditions—Specify values that a packet must contain to be considered a match.

- Action—Specifies what to do if a packet matches the match conditions. A filter can accept, discard, or reject a matching packet and then perform additional actions, such as counting, classifying, and policing. If no action is specified for a term, the default is to accept the matching packet.

Firewall Filter Processing

If there are multiple terms in a filter, the order of the terms is important. If a packet matches the first term, the switch executes the action defined by that term, and no other terms are evaluated. If the switch does not find a match between the packet and the first term, it compares the packet to the next term. If no match occurs between the packet and the second term, the system continues to compare the packet to each successive term in the filter until a match is found. If the packet does not match any terms in the filter, the switch discards the packet by default.

RELATED DOCUMENTATION

- Understanding Firewall Filter Planning | 853
- Understanding How Firewall Filters Are Evaluated | 854
- Understanding Firewall Filter Processing Points for Bridged and Routed Packets | 1853
- Understanding Firewall Filter Match Conditions | 848
- Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | 1731
- Firewall Filter Match Conditions and Actions (QFX10000 Switches) | 1780
- Overview of Policers | 2198
- Configuring Firewall Filters | 1826

Understanding Firewall Filter Match Conditions

IN THIS SECTION

- Filter Match Conditions | 849
- Numeric Filter Match Conditions | 849
- Interface Filter Match Conditions | 850
- IP Address Filter Match Conditions | 851
- Bit-Field Filter Match Conditions | 851

Before you define terms for firewall filters, you must understand how the conditions in a term are handled and how to specify interface, numeric, address, and bit-field filter match conditions to achieve the desired filter results.

Filter Match Conditions

In the `from` statement of a *firewall filter* term, you specify the conditions that the packet must match for the action in the `then` statement to be taken. All conditions must match for the action to be implemented. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify multiple values for the same condition, a match on any one of those values matches that condition. For example, if you specify multiple IP source addresses using the `source-address` statement, a packet that contains any one of those IP source addresses matches the condition. In some cases you can specify multiple values for the same condition by enclosing the possible values in square brackets, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set protocol (icmp | udp)
```

In other cases you must enter multiple statements, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-address 10.1.1.1
user@switch# set source-address 10.1.1.2
```

If you specify no match conditions in a term, that term matches all packets.

NOTE: Unlike traditional Junos OS firewall filters, you cannot use `except` in a condition statement to negate the condition.

Numeric Filter Match Conditions

You can specify numeric filter match conditions that are identified by a numeric value, such as port and protocol numbers. For numeric filter match conditions, you specify the condition and a single value that a field in a packet must contain to be considered a match.

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

- Single number—A match occurs if the value of the field matches the number. For example, to match Telnet traffic:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-port 23
```

- 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, to match Telnet traffic:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-port telnet
```

- To specify multiple values for the same match condition in a filter term, enter each value in its own match statement. For example, a match occurs in the following term if the value of the source port in the packet is 22 or 23.

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-port 22
user@switch# set source-port 23
```

Interface Filter Match Conditions

You can specify an interface filter match condition to match an interface on which a packet is received or transmitted. In this example, the final character (`0`) specifies the logical unit. You can include the wildcard (`*`) as part of the interface name. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/*/6.0
user@switch# set interface ge-0/1/*.0
user@switch# set interface ge-0/0/6.*
```

Note that you must specify a value or a wildcard for the logical unit.

IP Address Filter Match Conditions

You can specify an address filter match condition to match an IP source or destination address or prefix in a packet. Specify the address or prefix type and the address or prefix itself. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-address 10.2.1.0/24;
```

If you omit the prefix length, it defaults to /32. For example:

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

To specify more than one IP address or prefix in a filter term, enter each address or prefix in its own match statement. For example, a match occurs in the following term if the source address of a packet matches either of the following prefixes:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-address 10.1.0.0/16
user@switch# set source-address 10.2.0.0/16
```

Bit-Field Filter Match Conditions

You can specify bit-field filter match conditions to match particular bits within certain fields in Ethernet frames and IP, TCP, UDP, and ICMP headers. You usually specify the field and the bit within the field that must be set in a packet to be considered a match.

In most cases you can use a keyword to specify the bit you want to match on. For example, to match on a TCP SYN packet you can enter syn, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags syn
```

You can also enter `0x02` because the SYN bit is the third least-significant bit of the 8-bit tcp-flags field:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags 0x02
```

To match multiple bit-field values, use the logical operators, which are described in Table 34 on page 852. The operators are listed in order from highest precedence to lowest precedence. Operations are evaluated from left to right.

Table 34: Actions for Firewall Filters

Logical Operators	Description
!	Negation
&	Logical AND
|	Logical OR

If you use a logical operator, enclose the values in quotation marks and do not include any spaces. For example, the following statement matches the second packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn&ack"
```

To negate a match, precede the value with an exclamation point. For example, the following statement matches only the initial packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn&!ack"
```

You can use text synonyms to specify some common bit-field matches. For example, the following statement also matches the initial packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-initial
```

RELATED DOCUMENTATION

Understanding How a Firewall Filter Tests a Protocol | 1852

Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | 1731

Configuring Firewall Filters | 1826

Understanding Firewall Filter Planning

Before you create a *firewall filter* and apply it, determine what you want the filter to accomplish and how to use its match conditions and actions to achieve your goals. It is important that you understand how packets are matched, the default and configured actions of the firewall filter, and where to apply the firewall filter.

You can apply no more than one firewall filter per router interface per direction (input and output). For example, for a given interface you can apply at most one filter in the input direction and one filter in the output direction. You should try to be conservative in the number of terms (rules) that you include in each firewall filter, because a large number of terms requires longer processing time during a commit operation and can make testing and troubleshooting more difficult.

Before you configure and apply firewall filters, answer the following questions for each of them:

1. What is the purpose of the filter?

   For example, the system can drop packets based on header information, rate-limit traffic, classify packets into forwarding classes, log and count packets, or prevent denial-of-service attacks.

2. What are the appropriate match conditions? Determine the packet header fields that the packet must contain for a match. Possible fields include:

   - Layer 3 header fields—Source and destination IP addresses, protocols, and IP options (IP precedence, IP fragmentation flags, or TTL type).
   - TCP header fields—Source and destination ports and flags.
   - ICMP header fields—Packet type and code.

3. What are the appropriate actions to take if a match occurs?

   The system can accept, discard, or reject packets.

4. What additional action modifiers might be required?

   For example, you can configure the system to mirror (copy) packets to a specified port, count matching packets, apply traffic management, or police packets.

5. On what Layer 3 interface should the firewall filter be applied?

   Before you choose the interface on which to apply a firewall filter, understand how that placement can affect traffic flow to other interfaces. In general, apply a filter close to the source device if the filter matches on source or destination IP addresses, IP protocols, or protocol information—such as ICMP message types, and TCP or UDP port numbers. However, you should apply a filter close to the destination device if the filter matches *only* on a source IP address. When you apply a filter too close to the source device, the filter could prevent that source device from accessing other services that are available on the network.

6. In which direction should the firewall filter be applied?

   You typically configure different actions for traffic entering an interface than you configure for traffic exiting an interface.

7. How many filters should I create?

RELATED DOCUMENTATION

Overview of Policers | **2198**

Understanding How Firewall Filters Are Evaluated | **854**

Configuring Firewall Filters | **1826**

Understanding How Firewall Filters Are Evaluated

A *firewall filter* consists of one or more terms, and the order of the terms within a filter is important. Before you configure firewall filters, you should understand how switches evaluate the terms within a filter and how packets are evaluated against the terms.

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 no action is specified in the `then` statement, the default action **accept** is taken.
- If the packet does not match all the conditions, the switch discards it.

When a firewall filter consists of more than one term, the filter is evaluated sequentially:

1. The packet is evaluated against the conditions in the `from` statement in the first term.

2. If the packet matches all the conditions in the term, the action in the `then` statement is taken and the evaluation ends. Subsequent terms in the filter are not evaluated.

3. If the packet does not match all the conditions in the term, the packet is evaluated against the conditions in the `from` statement in the second term.

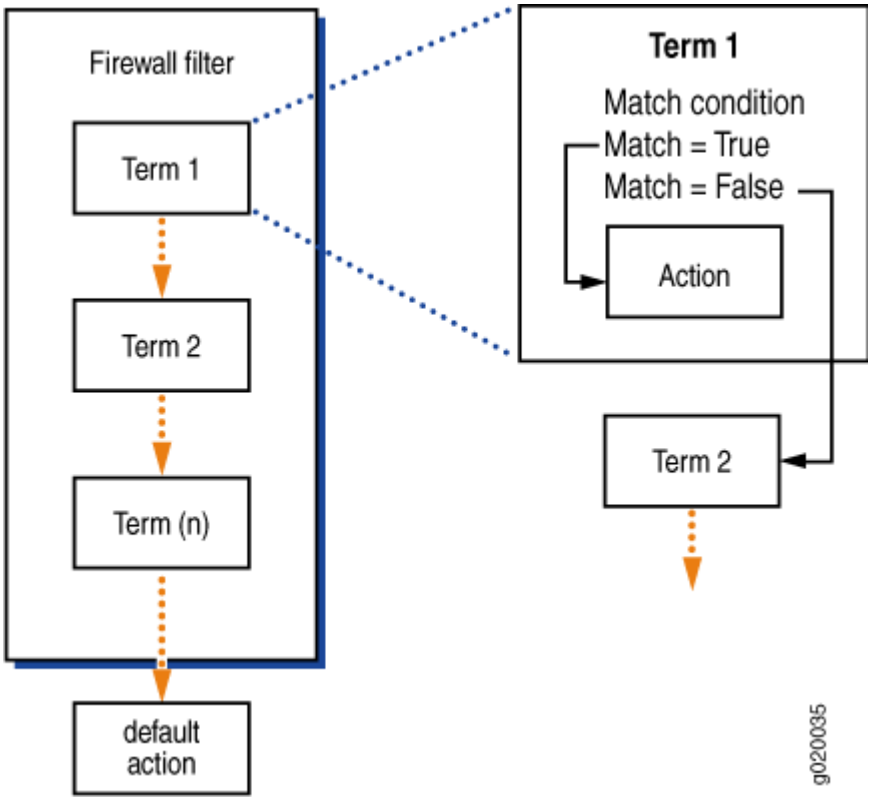
   This process continues until the packet matches all the conditions in the `from` statement in one of the subsequent terms or there are no more terms in the filter.

4. If a packet passes through all the terms in the filter without a match, the switch discards it.

NOTE: The order of conditions in a `from` statement is not important because a packet must match all the conditions to be considered a match.

Figure 50 on page 855 shows how switches evaluate the terms within a firewall filter.

Figure 50: Evaluation of Terms Within a Firewall Filter



If you do not include a `from` statement in a term, all packets will match the term and be processed by the `then` statement. If a term does not contain a `then` statement or if an action has not been configured in the `then` statement, the term accepts any matching packets.

Every firewall filter contains an implicit `deny` statement at the end of the filter, which is equivalent to the following explicit filter term:

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

Consequently, a packet that does not match any of the terms in a firewall filter is discarded. If you configure a filter that has no terms, all packets that pass through the filter are discarded.

NOTE: Firewall filtering is supported on packets that are at least 64 bytes long.

RELATED DOCUMENTATION

- Understanding Firewall Filter Match Conditions | **856**
- Overview of Policers | **2198**
- Configuring Firewall Filters | **1826**

Understanding Firewall Filter Match Conditions

IN THIS SECTION

- Filter Match Conditions | **857**
- Numeric Filter Match Conditions | **857**
- Interface Filter Match Conditions | **858**
- IP Address Filter Match Conditions | **859**
- MAC Address Filter Match Conditions | **859**
- Bit-Field Filter Match Conditions | **860**

Before you define terms for firewall filters, you must understand how the conditions in a term are handled and how to specify interface, numeric, address, and bit-field filter match conditions to achieve the desired filter results.

Filter Match Conditions

In the `from` statement of a *firewall filter* term, you specify the conditions that the packet must match for the action in the `then` statement to be taken. All conditions must match for the action to be implemented. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify multiple values for the same condition, a match on any one of those values matches that condition. For example, if you specify multiple IP source addresses using the `source-address` statement, a packet that contains any one of those IP source addresses matches the condition. In some cases you can specify multiple values for the same condition by enclosing the possible values in square brackets, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set protocol (icmp | udp)
```

In other cases you must enter multiple statements, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-address 10.1.1.1
user@switch# set source-address 10.1.1.2
```

If you specify no match conditions in a term, that term matches all packets.

NOTE: Unlike traditional Junos OS firewall filters, you cannot use `except` in a condition statement to negate the condition.

Numeric Filter Match Conditions

You can specify numeric filter match conditions that are identified by a numeric value, such as port and protocol numbers. For numeric filter match conditions, you specify the condition and a single value that a field in a packet must contain to be considered a match.

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

- Single number—A match occurs if the value of the field matches the number. For example, to match Telnet traffic:

  ```
  [edit firewall family family-name filter filter-name term term-name from]
  user@switch# set source-port 23
  ```

- 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, to match Telnet traffic:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-port telnet
```

- To specify multiple values for the same match condition in a filter term, enter each value in its own match statement. For example, a match occurs in the following term if the value of the source port in the packet is 22 or 23.

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-port 22
user@switch# set source-port 23
```

Interface Filter Match Conditions

You can specify an interface filter match condition to match an interface on which a packet is received or transmitted. For example, if you apply a filter to a VLAN you might want the filter to match on some interfaces that participate in the VLAN and not match on other interfaces in the VLAN. When you specify the name of the interface, you must include a logical unit.

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/0/6.0
```

In this example, the final character (`0`) specifies the logical unit. You can include the wildcard (`*`) as part of the interface name. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/*/6.0
user@switch# set interface ge-0/1/*.0
user@switch# set interface ge-0/0/6.*
```

Note that you must specify a value or a wildcard for the logical unit.

IP Address Filter Match Conditions

You can specify an address filter match condition to match an IP source or destination address or prefix in a packet. Specify the address or prefix type and the address or prefix itself. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-address 10.2.1.0/24;
```

If you omit the prefix length, it defaults to /32. For example:

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

To specify more than one IP address or prefix in a filter term, enter each address or prefix in its own match statement. For example, a match occurs in the following term if the source address of a packet matches either of the following prefixes:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-address 10.1.0.0/16
user@switch# set source-address 10.2.0.0/16
```

MAC Address Filter Match Conditions

You can specify a MAC address filter match condition to match a source or destination MAC address. You specify the address type and value that a packet must contain to be considered a match.

You can specify the MAC address as six hexadecimal bytes in any of the following formats:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 00:11:22:33:44:55
```

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 0011.2233.4455
```

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 001122334455
```

Regardless of the formats you use, the system resolves the address to the standard format, in this case 00:11:22:33:44:55.

To specify more than one MAC address in a filter term, enter each MAC address in its own match statement. For example, a match occurs in the following term if the value of the MAC source address matches either of the following addresses:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-mac-address 00:11:22:33:44:55
user@switch# set source-mac-address 00:11:22:33:20:15
```

Bit-Field Filter Match Conditions

You can specify bit-field filter match conditions to match particular bits within certain fields in Ethernet frames and IP, TCP, UDP, and ICMP headers. You usually specify the field and the bit within the field that must be set in a packet to be considered a match.

In most cases you can use a keyword to specify the bit you want to match on. For example, to match on a TCP SYN packet you can enter `syn`, as in:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags syn
```

You can also enter `0x02` because the SYN bit is the third least-significant bit of the 8-bit tcp-flags field:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags 0x02
```

To match multiple bit-field values, use the logical operators, which are described in Table 35 on page 861. The operators are listed in order from highest precedence to lowest precedence. Operations are evaluated from left to right.

Table 35: Actions for Firewall Filters

Logical Operators	Description
!	Negation
&	Logical AND
|	Logical OR

If you use a logical operator, enclose the values in quotation marks and do not include any spaces. For example, the following statement matches the second packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn&ack"
```

To negate a match, precede the value with an exclamation point. For example, the following statement matches only the initial packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn&!ack"
```

You can use text synonyms to specify some common bit-field matches. For example, the following statement also matches the initial packet of a TCP handshake:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-initial
```

RELATED DOCUMENTATION

Understanding How a Firewall Filter Tests a Protocol | 1852

Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700) | 1732

Configuring Firewall Filters | 1826

Firewall Filter Flexible Match Conditions

IN THIS SECTION

- Statement Hierarchy | 863
- Flexible Filter Match Types | 863
- Flexible Filter Match Start Locations | 865
- Flexible Filter Match Examples | 866

Standard firewall filter match conditions vary based on the protocol family of the traffic being matched. For example, the terms available for bridge protocol traffic are different from those available for the inet or inet6 protocol families. The fields available for matching within each protocol family are, however, fixed or pre-defined. This means that filters can match on patterns within those pre-defined fields only.

Using flexible match conditions, firewall filters can be constructed that start the match at layer-2, layer-3, layer-4 or payload locations. From there, additional offset criteria can be specified thereby enabling pattern matches at custom, user-defined locations within a packet.

Flexible match filter terms are applied to MPC or MIC interfaces as either input or output filters just as any other firewall filter terms. Flexible match filter terms can also be created as templates at the `[edit firewall]` hierarchy level. These templates can then be referenced within a flexible match term.

For MX series routers, flexible match conditions are only supported with MPCs or MICs. For environments where FPCs, PICs, and or DPCs are installed along with MPCs or MICs, be sure to only apply the flexible match firewall filter criteria to the MPC or MIC interfaces.

NOTE: For MX Series routers with MPCs, you need to initialize the filter counter for Trio-only match filters in the MIB by walking the corresponding SNMP MIB. For example, for any filter that is configured or changed with respect to their Trio-only filters, you need to run a command such

as the following: `show snmp mib walk (ascii | decimal) object-id`. This forces Junos to learn the filter counters and ensure that the filter statistics are displayed (this is because the first poll to filter statistics may not show all counters). Trio-only match filters are those that include at least one match condition or action that is only supported by the Trio chipset.

This guidance applies to all `enhanced-mode` firewall filters. It also applies to "Firewall Filter Match Conditions for IPv4 Traffic" on page 940 with flexible match filter terms for offset-range or offset-mask, `gre-key`, and " Firewall Filter Match Conditions for IPv6 Traffic" on page 957 with any of the following match conditions: `payload-protocol`, `extension headers`, `is_fragment`. It also applies to filters with either of the following "Firewall Filter Terminating Actions" on page 884: `encapsulate` or `decapsulate`, or either of the following "Firewall Filter Nonterminating Actions" on page 871: `policy-map`, and `clear-policy-map`.

Statement Hierarchy

Flexible match filter terms are available in three variations as shown in Table 36 on page 863. The `flexible-match` variation is configured at the `[edit firewall]` hierarchy level. It is used to define flexible match templates. The `flexible-filter-match-mask` and `flexible-match-range` are configured at the `[edit firewall family [inet|inet6|bridge|ethernet-switching|ccc|vpls] filter <filter-name> term <term-name> from]` hierarchy. Use the `family ethernet-switching` filter for EX9200 switches.

Flexible Filter Match Types

Table 36: Flexible Filter Match Types

Flexible Filter Match Type	Available Attributes	Description
`flexible-match`	*<name>*	Create a flexible-match template named as the *<name>* attribute.
	`bit-length`	Length of the data to be matched in bits, not needed for string input (0..32) For QFX5120 and EX4650 switches, 16 and 32 are the only valid bit lengths.
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)

Table 36: Flexible Filter Match Types *(Continued)*

Flexible Filter Match Type	Available Attributes	Description
	byte-offset	Byte offset after the match start point
	match-start	Start point to match in packet
flexible-match-mask	bit-length	Length of the data to be matched in bits, not needed for string input (0..128)
	bit-offset	Bit offset after the (match-start + byte) offset (0..7)
	byte-offset	Byte offset after the match start point
	flexible-mask-name	Select a flexible match from predefined template field. Required unless match-start is configured.
	mask-in-hex	Mask out bits in the packet data to be matched.
	match-start	Start point to match in packet. Required unless flexible-mask-name is configured.
	prefix	Value data/string to be matched.
flexible-match-range	bit-length	Length of the data to be matched in bits. (0..32) Required unless flexible-range-name is configured.
	bit-offset	Bit offset after the (match-start + byte) offset. (0..7)
	byte-offset	Byte offset after the match start point
	flexible-range-name	Select a flexible match from predefined template.

Table 36: Flexible Filter Match Types *(Continued)*

Flexible Filter Match Type	Available Attributes	Description
	`match-start`	Start point to match in packet. Required unless `flexible-range-name` is configured.
	`range`	Range of values to be matched.
	`range-except`	Range of values to be not matched.

Flexible Filter Match Start Locations

Flexible match filter terms are constructed by giving a start location or anchor point within the packet. The start locations can be any of: layer-2, layer-3, layer-4 or payload, depending on the protocol family in use. Table 37 on page 865 shows available flexible filter match start locations by protocol family. You use these available start locations as the `match-start` locations for the flexible match filter terms.

From these start locations, specific byte and bit offsets can be utilized to allow the filter to match patterns at very specific locations within the packet.

Table 37: Flexible Filter Match Start Locations

Protocol Family	Available Start Locations
`inet`	`layer-3, layer-4 and payload` For QFX5120 and EX4650 switches, support for layer-2 and layer-3 (only) flexible match filters was added in Junos Release 20.1R1.
`inet6`	`layer-3, layer-4 and payload` For QFX5120 and EX4650 switches, support for layer-2 and layer-3 (only) flexible match filters was added in Junos Release 20.1R1.
`bridge`	`layer-2, layer-3, layer-4 and payload`

Table 37: Flexible Filter Match Start Locations *(Continued)*

Protocol Family	Available Start Locations
ccc	layer-2, layer-3, layer-4 and payload
mpls	layer-3 and payload
vpls	layer-2, layer-3, layer-4 and payload
ethernet-switching	(EX9200 switches) layer-2, layer-3, layer-4 and payload For, QFX5120 and EX4650 switches, support for layer-2 and layer-3 (only) flexible match filters was added in Junos Release 20.1R1. An example of using a layer-2 packet offset and match length can be found below.

Flexible Filter Match Examples

The following example illustrates the use and context for `flexible-match-mask`.

```
from {
    flexible-match-mask {
        flexible-mask-name <mask-name>;
        mask-in-hex <mask>;
        prefix <pattern>;
     }
}
```

The *<mask-name>* specifies for *flexible-mask-name* which predefined template is used for the flexible match condition. Templates can be defined to specify at which place (position) in the packet the flexible match condition should be executed.

The *<mask>* for *mask-in-hex* is in hexadecimal format. For example, a configured mask of `0xf0fc` specifies a match for the fist four bits in first byte (as referred by *<mask-name>*), and for the first six bits in the second byte. If the packet is IPv4 packet, and *<mask-name>* refers to first two bytes in L3 header, the search is for the IP version field and DSCP field. As another example, a configured mask `0xffc0` specifies a search for entire first byte and for two bits from the second byte. If the *<mask-name>* refers to first two bytes in L3 header, and the packet is IPv6 packet, this specifies the IP version field and DSCP in the Traffic Class field.

The *<pattern>* specified for *prefix* is an ASCII string. If first two characters are `0x`, then the string is processed as a hexadecimal number encoding appropriate bits. For example, the configured prefix `0x40c0` in combination with mask `0xf0fc` and *<mask-name>* referring first two bytes in L3 header, indicates a search for `0100` in the first four bits (version field is equal to 4) and `1100 00` in IPv4 DSCP field (DSCP is equal to cs6). Or, using the configured prefix `0x6c00` in combination with mask `0xffc0` and *<mask-name>* referring first two bytes in L3 header, specifies a search for for `0110` in the first four bits (version field is equal to 6), and `1100 00` in IPv6 DSCP field (DSCP is equal to cs6).

The first example defines a mask template that selects first two bytes (16 bits) from L3 header for flexible match:

```
firewall {
    flexible-match FM-FIRST-TWO-L3-BYTES {
        match-start layer-3;
        byte-offset 0;
        bit-offset 0;
        bit-length 16;
    }
}
```

The next example defines a mask template that selects the third through sixth byte (32 bits) of the packet payload for flexible match:

```
firewall {
    flexible-match FM-FOUR-PAYLOAD-BYTES {
        match-start payload;
        byte-offset 2;
        bit-offset 0;
        bit-length 32;
    }
}
```

This example shows an ASCII character match for the string *JNPR* (ASCII characters: `0x4a`, `0x4e`, `0x50`, `0x52`) in the third through sixth byte of the packet payload. The filter uses the `FM-FOUR-PAYLOAD-BYTES` mask template defined in the previous example.

```
firewall {
    family ccc filter FF-COUNT-JNPR-PACKETS {
        term JNPR-STRING {
            from {
                flexible-match-mask {
```

```
                    mask-in-hex 0xffffffff;
                    prefix JNPR;
                    flexible-mask-name FM-FOUR-PAYLOAD-BYTES;
                }
            }
            then {
                count CNT-JNPR-YES
                accept;
            }
        }
        term DEAFULT {
            then {
                count CNT-JNPR-NO
                accept;
            }
        }
    }
}
```

This example shows a family ccc filter looking for DSCP equal to `cs6` and DSCP `ef`, regardless whether the encapsulated packets are IPv4 or IPv6. It uses the the `FM-FIRST-TWO-L3-BYTES` mask template defined in the first example.

```
firewall {
    family ccc filter FF-DSCP-CLASSIFY {
        term ROUTING-IPV4 {
            from {
                flexible-match-mask {
                    mask-in-hex 0xf0fc;
                    prefix 0x40c0;        # DSCP=cs6 in IPv4 header
                    flexible-mask-name FM-FIRST-TWO-L3-BYTES;
                }
            }
            then {
                count ROUTING-IPV4;
                accept;
            }
        }
        term ROUTING-IPV6 {
            from {
                flexible-match-mask {
                    mask-in-hex 0xffc0;
```

```
                    prefix 0x6c00;        # DSCP=cs6 in IPv6 header
                    flexible-mask-name FM-FIRST-TWO-L3-BYTES;
                }
            }
            then {
                count ROUTING-IPV6;
                accept;
            }
        }
        term VOICE-IPV4 {
            from {
                flexible-match-mask {
                    mask-in-hex 0xf0fc;
                    prefix 0x40b8;            # DSCP=ef in IPv4 header
                    flexible-mask-name FM-FIRST-TWO-L3-BYTES;
                }
            }
            then {
                count VOICE-IPV4;
                accept;
            }
        }
        term VOICE-IPV6 {
            from {
                flexible-match-mask {
                    mask-in-hex 0xffc0;
                    prefix 0x6b80;             # DSCP=ef in IPv6 header
                    flexible-mask-name FM-FIRST-TWO-L3-BYTES;
                }
            }
            then {
                count VOICE-IPV6;
                accept;
            }
        }
        term DEFAULT {
            then {
                accept;
            }
        }
    }
}
```

This example shows how to use a match length, starting from a layer-2 packet offset, in a firewall filter for a QFX5120-32C, QFX5120-48Y, or EX4650 device running Junos Release 20.1R1. Here, we use a bit-length of 32 bits and the `ethernet-switching` family (`inet` and `inet6` are also supported, as is using a layer-3 offset).

```
user@device# show firewall family ethernet-switching
filter udf_eth {
    term t1 {
        from {
            flexible-match-mask {
                match-start layer-2;
                byte-offset 8;
                bit-length 32;
                prefix 168430090;
            }
        }
        then count c1;
    }
}
```

Release History Table

Release	Description
23.2R1	From Junos OS Release 23.2R1, we support layer-2 and layer-3 (only) flexible match filters on the EX4100-24P, EX4100-24T, EX4100-24MP, EX4100-48P, EX4100-48T, EX4400-24T, EX4400-24X, and EX4400-48F switches.
20.1R1	For, QFX5120 and EX4650 switches, support for layer-2 and layer-3 (only) flexible match filters was added in Junos Release 20.1R1.

RELATED DOCUMENTATION

- Firewall Filter Match Conditions for IPv4 Traffic | **940**
- Firewall Filter Match Conditions for IPv6 Traffic | **957**
- Firewall Filter Match Conditions for Layer 2 CCC Traffic | **1032**
- Firewall Filter Match Conditions for VPLS Traffic | **1013**

Firewall Filter Nonterminating Actions

Firewall filters support different sets of nonterminating actions for each protocol family, which include an implicit accept action. In this context, *nonterminating* means that other actions can follow these actions whereas no other actions can follow a *terminating* action. As such, you cannot configure the `next term` action with a *terminating* action in the same filter term. You can, however, configure the `next term` action with another *nonterminating* action in the same filter term.

> **NOTE**: On Junos OS and Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

Table 38 on page 872 describes the nonterminating actions you can configure for a firewall filter term.

Table 38: Nonterminating Actions for Firewall Filters

Nonterminating Action	Description	Protocol Families
`bgp-output-queue-priority priority` *`(expedited | (1-16))`*	Assign the packet to one of the 17 prioritized BGP output queues.	• `family evpn` • `family inet` • `family inet-mdt` • `family inet-mvpn` • `family inet-vpn` • `family inet6` • `family inet6-mvpn` • `family inet6-vpn` • `family iso-vpn` • `family l2vpn` • `family route-target` • `family traffic-engineering`
`count` *`counter-name`*	Count the packet in the named counter.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`dont-fragment (set | clear)`	Configure the value of the Don't Fragment bit (flag) in the IPv4 header to specify whether the datagram can be fragmented: • `set`—Change the flag value to one, preventing fragmentation. • `clear`—Change the flag value to zero, allowing fragmentation. **NOTE**: The `dont-fragment (set | clear)` actions are supported only on MPCs.	`family inet`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`dscp` *value*	Set the IPv4 Differentiated Services code point (DSCP) bit. You can specify a numerical value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. The default DSCP value is `be` (best effort), or `0`. You can also specify one of the following text synonyms: • `af11`—Assured forwarding class 1, low drop precedence (1) • `af12`—Assured forwarding class 1, medium drop precedence (2) • `af13`—Assured forwarding class 1, high drop precedence (3); and so on through `af43`, Assured forwarding class 4, high drop precedence • `be`—Best effort • `cs0`—Class selector 0; and so on through `cs7`, Class selector 0 • `ef`—Expedited forwarding **NOTE**: This action is not supported on PTX series routers. **NOTE**: MPC line cards running on MX series routers support any value (from 0 to 63) in conjunction with the `set dscp` firewall filter action. **NOTE**: The actions `dscp 0` and `dscp be` are supported only on T320, T640, T1600, TX Matrix, TX Matrix Plus, and M320 routers and on 10-Gigabit Ethernet Modular Port Concentrators (MPC). However, these actions are not supported on Enhanced III Flexible PIC Concentrators (FPCs) on M320 routers. On T4000 routers, the `dscp 0` action is not supported during the inter-operation between a T1600 Enhanced Scaling Type 4 FPC and a T4000 Type 5 FPC.	`family inet`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`enhanced-hierarchical-policer`	Police the packets of a traffic priority using the specified enhanced hierarchical policer.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family VPLS`
`force-premium`	By default, a hierarchical policer processes the traffic it receives according to the traffic's forwarding class. Premium, expedited-forwarding traffic, has priority for bandwidth over aggregate, best-effort traffic. The `force-premium` filter ensures that traffic matching the term is treated as premium traffic by a subsequent hierarchical policer, regardless of its forwarding class. This traffic is given preference over any aggregate traffic received by that policer. **NOTE**: The `force-premium` filter option is supported only on MPCs.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family VPLS`
`forwarding-class` *`class-name`*	Classify the packet to the named forwarding class: • *forwarding-class-name* • `assured-forwarding` • `best-effort` • `expedited-forwarding` • `network-control`	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`hierarchical-policer`	Police the packet using the specified hierarchical policer	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`
`ipsec-sa` *ipsec-sa*	Use the specified IPsec security association. **NOTE**: This action is not supported on MX Series routers, Type 5 FPCs on T4000 routers, and PTX Series Packet Transport Routers.	`family inet`
`load-balance` *group-name*	Use the specified load-balancing group. **NOTE**: This action is not supported on MX Series routers or PTX Series Packet Transport Routers.	`family inet`
`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 at the command-line interface (CLI). **NOTE**: The Layer 2 (L2) families log action is available only for MX Series routers with MPCs (MPC mode if the router has only MPCs, or mix mode if it has MPCs and DCPs). For MX Series routers with DPCs, the log action for L2 families is ignored if configured.	• `family bridge` • `family ccc` • `family inet` • `family inet6` • `family vpls`
`logical-system` *logical-system-name*	Direct packets to a specific logical system.	• `family inet` • `family inet6`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`loss-priority (high | medium-high | medium-low | low)`	Set the packet loss priority (PLP) level. You cannot also configure the `three-color-policer` nonterminating action for the same firewall filter term. These two nonterminating actions are mutually exclusive. This action is supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement and using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`
`next-hop-group` *`group-name`*	Use the specified next-hop group. We recommend that you do not use the `next-hop-group` action with the `port-mirror-instance` or `port-mirror` action in the same firewall filter.	• `family any` • `family inet`
`next-interface` *`interface-name`*	(MX Series) Direct packets to the specified outgoing interface.	• `family inet` • `family inet6`
`next-ip` *`ip-address`*	(MX Series) Direct packets to the specified destination IPv4 address.	`family inet`
`next-ip6` *`ipv6-address`*	(MX Series) Direct packets to the specified destination IPv6 address.	`family inet6`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`packet-mode`	Updates a bit field in the packet key buffer, which specifies traffic that will bypass flow-based forwarding. Packets with the `packet-mode` action modifier follow the packet-based forwarding path and bypass flow-based forwarding completely. Applies to SRX100, SRX210, SRX220, SRX240, and SRX650 devices only. For more information about selective stateless packet-based services, see the *Junos OS Security Configuration Guide*.	`family any`
`policer` *`policer-name`*	Name of policer to use to rate-limit traffic.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`
`policy-map` *`policy-map-name`*	(MX Series) Name of policy map used to assign specific rewrite rules to a specific customer.	• `family any` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`port-mirror` *instance-name*	Port-mirror the packet based on the specified family. This action is supported on M120 routers, M320 routers configured with Enhanced III FPCs, MX Series routers, and PTX Series Packet Transport Routers only. We recommend that you do not use both the `next-hop-group` and the `port-mirror` actions in the same firewall filter.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family vpls` • `family mpls`
`port-mirror-instance` *instance-name*	Port mirror a packet for an instance. This action is supported only on the MX Series routers. We recommend that you do not use both the `next-hop-group` and the `port-mirror-instance` actions in the same firewall filter.	• `family any` • `family bridge` • `family ccc` • `family inet` • `family inet6` • `family vpls` • `family mpls`
`prefix-action` *action-name*	Count or police packets based on the specified action name. **NOTE**: This action is not supported on PTX Series Packet Transport Routers.	`family inet`
`routing-instance` *routing-instance-name*	Direct packets to the specified routing instance.	• `family inet` • `family inet6` •

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`sample`	Sample the packet. **NOTE**: Junos OS does not sample packets originating from the router. If you configure a 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.	• `family inet` • `family inet6` • `family mpls`
`service-accounting`	Use the inline counting mechanism when capturing subscriber per-service statistics. Count the packet for service accounting. The count is applied to a specific named counter (`__junos-dyn-service-counter`) that RADIUS can obtain. The `service-accounting` and `service-accounting-deferred` keywords are mutually exclusive, both per-term and per-filter. **NOTE**: This action is not supported on T4000 Type 5 FPCs and PTX Series Packet Transport Routers.	• `family any` • `family inet` • `family inet6`
`service-accounting-deferred`	Use the deferred counting mechanism when capturing subscriber per-service statistics. The count is applied to a specific named counter (`__junos-dyn-service-counter`) that RADIUS can obtain. The `service-accounting` and `service-accounting-deferred` keywords are mutually exclusive, both per-term and per-filter. **NOTE**: This action is not supported on T4000 Type 5 FPCs and PTX Series Packet Transport Routers.	• `family any` • `family inet` • `family inet6`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`service-filter-hit`	(Only if the `service-filter-hit` flag is marked by a previous filter in the current type of chained filters) Direct the packet to the next type of filters. Indicate to subsequent filters in the chain that the packet was already processed. This action, coupled with the `service-filter-hit` match condition in receiving filters, helps to streamline filter processing. **NOTE**: This action is not supported on T4000 Type 5 FPCs and PTX Series Packet Transport Routers.	• `family any` • `family inet` • `family inet6`
`slice` *`slice-name`*	Mark packets that pass the match conditions of the rule with the slice identifier corresponding to the Services Network-Slicing configuration. See slice (firewall filter action).	• `family any` • `family bridge` • `family ccc` • `family evpn` • `family inet` • `family inet6` • `family mpls` • `family vpls`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`syslog`	Log the packet to the system log file. The syslog firewall action for existing `inet` and `inet6` families, and the `syslog` action in L2 family filters includes the following L2 information: Input interface, action, VLAN ID1, VLAN ID2, Ethernet type, source and destination MAC addresses, protocol, source and destination IP addresses, source and destination ports, and the number of packets. **NOTE**: The L2 families syslog action is available only for MX Series routers with MPCs (MPC mode if the router has only MPCs, or mix mode if it has MPCs and DCPs). For MX Series routers with DPCs, the syslog action for L2 families is ignored if configured.	• `family bridge` • `family ccc` • `family inet` • `family inet6` • `family vpls`
`three-color-policer (single-rate | two-rate)` *`policer-name`*	Police the packet using the specified single-rate or two-rate three-color-policer. **NOTE**: You cannot also configure the `loss-priority` action for the same firewall filter term. These two actions are mutually exclusive.	• `family bridge` • `family ccc` • `family inet` • `family inet6` • `family mpls` • `family vpls`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
`traffic-class` *value*	Specify the traffic-class code point. You can specify a numerical value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. The default traffic-class value is best effort, that is, `be` or `0`. In place of the numeric value, you can specify one of the following text synonyms: • `af11`—Assured forwarding class 1, low drop precedence • `af12`—Assured forwarding class 1, medium drop precedence • `af13`—Assured forwarding class 1, high drop precedence • `af21`—Assured forwarding class 2, low drop precedence • `af22`—Assured forwarding class 2, medium drop precedence • `af23`—Assured forwarding class 2, high drop precedence • `af31`—Assured forwarding class 3, low drop precedence • `af32`—Assured forwarding class 3, medium drop precedence • `af33`—Assured forwarding class 3, high drop precedence • `af41`—Assured forwarding class 4, low drop precedence • `af42`—Assured forwarding class 4, medium drop precedence • `af43`—Assured forwarding class 4, high drop precedence • `be`—Best effort • `cs0`—Class selector 0 • `cs1`—Class selector 1 • `cs2`—Class selector 2	`family inet6`

Table 38: Nonterminating Actions for Firewall Filters *(Continued)*

Nonterminating Action	Description	Protocol Families
	• `cs3`—Class selector 3 • `cs4`—Class selector 4 • `cs5`—Class selector 5 • `cs6`—Class selector 6 • `cs7`—Class selector 7 • `ef`—Expedited forwarding **NOTE**: The actions `traffic-class 0`and `traffic-class be` are supported only on T Series and M320 routers and on the 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Ethernet Queuing MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers. However, these actions are not supported on Enhanced III Flexible PIC Concentrators (FPCs) on M320 routers.	

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters | 814

Firewall Filter Terminating Actions | 884

Firewall Filter Terminating Actions

Firewall filters support a set of terminating actions for each protocol family. A filter-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.

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

On Junos OS and Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.

For MX Series routers with MPCs, you need to initialize the filter counter for Trio-only match filters by walking the corresponding SNMP MIB, for example, `show snmp mib walk name ascii`. This forces Junos to learn the filter counters and ensure that the filter statistics are displayed. This guidance applies to all enhanced mode firewall filters, filters with flexible conditions, and filters with the certain terminating actions. See those topics, listed under Related Documentation, for details.

Table 39 on page 885 describes the terminating actions you can specify in a firewall filter term.

Table 39: Terminating Actions for Firewall Filters

Terminating Action	Description	Protocols
accept	Accept the packet.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`decapsulate gre [ routing-instance` *`instance-name`* `]`	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable de-encapsulation of generic routing encapsulation (GRE) packets transported through a filter-based GRE tunnel. You can configure a filter term that pairs this action with a match condition that includes a packet header match for the GRE protocol. For an IPv4 filter, include the `protocol gre` (or `protocol 47`) match condition. Attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a configuration that attaches a de-encapsulating filter	• `family inet`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	to an interface that does not support filter-based GRE tunneling, the system writes a syslog warning message that the interface does not support the filter. When the interface receives a matched packet, processes that run on the Packet Forwarding Engine perform the following operations: • Remove the outer GRE header. • Forward the inner payload packet to its original destination by performing destination lookup. By default, the Packet Forwarding Engine uses the default routing instance to forward payload packets to the destination network. If the payload is MPLS, the Packet Forwarding	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	Engine performs route lookup on the MPLS path routing table using the route label in the MPLS header. If you specify the **decapsulate** action with an optional routing instance name, the Packet Forwarding Engine performs route lookup on the routing instance, and the instance must be configured. **NOTE**: On MX960 routers, the `decapsulate` action de-encapsulates GRE, IP-in-IP and IPv6-in-IP tunneling packets. You configure this action at the [`edit firewall family inet filter` *`filter-name`* `term` *`term-name`*] hierarchy level . For more information, see "Understanding Filter-Based Tunneling Across IPv4 Networks" on page 1394 and "Components of	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	Filter-Based Tunneling Across IPv4 Networks" on page 1403.	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`decapsulate l2tp [ routing-instance` *`instance-name`* `] [ forwarding-class` *`class-name`* `] [ output-interface` *`interface-name`* `] [ cookie` *`l2tpv3-cookie`* `] [ sample ]`	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable de-encapsulation of Layer 2 tunneling protocol (L2TP) packets transported through a filter-based L2TP tunnel. You can configure a filter term that pairs this action with a match condition that includes a packet header match for the L2TP protocol. For IPv4 traffic, an input firewall filter `$junos-input-filter` and an output firewall filter `$junos-output-filter` are attached to the interface. Attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a	`family inet`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	configuration that attaches a de-encapsulating filter to an interface that does not support filter-based L2TP tunneling, the system writes a syslog warning message that the interface does not support the filter. The remote tunnel endpoint sends an IP tunnel packet that contains an Ethernet MAC address in the payload. If the destination MAC address of the payload packet contains the MAC address of the router, the Ethernet packet is sent in the outgoing direction towards the network, and it is processed and forwarded as though it is received on the customer port. If the source MAC address of the payload packet contains the MAC address of the router, the Ethernet packet is transmitted in the outgoing direction towards	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	the customer port. If the tunnel does not contain the receive-cookie configured, packet injection does not happen. In such a case, any received tunnel packet is counted and dropped in the same manner in which packets that arrive with a wrong cookie are counted and dropped. The following parameters can be specified with the `decapsulate l2tp` action: • `routing-instance` *`instance-name`*—By default, the Packet Forwarding Engine uses the default routing instance to forward payload packets to the destination network. If the payload is MPLS, the Packet Forwarding Engine performs route lookup on the MPLS path	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	routing table using the route label in the MPLS header. If you specify the `decapsulate` action with an optional routing instance name, the Packet Forwarding Engine performs route lookup on the routing instance, and the instance must be configured. • `forwarding-class` *`class-name`*—(Optional) Classify l2TP packets to the specified forwarding class. • `output-interface` *`interface-name`*—(Optional) For L2TP tunnels, enable the packet to be duplicated and sent towards the customer or the network (based on the MAC address in the Ethernet payload).	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	• `cookie` *`l2tpv3-cookie`*—(Optional) For L2TP tunnels, specify the L2TP cookie for the duplicated packets. If the tunnel does not contain the receive-cookie configured, packet injection does not happen. In such a case, any received tunnel packet is counted and dropped in the same manner in which packets that arrive with a wrong cookie are counted and dropped. • `sample`—(Optional) Sample the packet. Junos OS does not sample packets originating from the router. If you configure a filter and apply it to the output side of an interface,	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	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. **NOTE**: The `decapsulate l2tp` action that you configure at the `[edit firewall family inet filter` *`filter-name`* `term` *`term-name`*] hierarchy level does not process traffic with IPv4 and IPv6 options. As a result, traffic with such options is discarded by the de-encapsulation of L2TP packets functionality.	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`discard`	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are available for logging and sampling.	• `family any` • `family inet` • `family inet6` • `family mpls` • `family vpls` • `family ccc` • `family bridge` • **family ethernet-switching** (for EX Series switches only)

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`encapsulate` *`template-name`*	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable filter-based generic routing encapsulation (GRE) tunneling using the specified tunnel template. You can configure a filter term that pairs this action with the appropriate match conditions, and then attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a configuration that attaches an encapsulating filter to an interface that does not support filter-based GRE tunneling, the system writes a syslog warning	• `family inet` • `family inet6` • `family any` • `family mpls`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	message that the interface does not support the filter. When the interface receives a matched packet, processes that run on the Packet Forwarding Engine use information in the specified tunnel template to perform the following operations: 1. Attach a GRE header (with or without a tunnel key value, as specified in the tunnel template. 2. Attach a header for the IPv4 transport protocol. 3. Forward the resulting GRE packet from the tunnel source interface to the tunnel destination (the remote PE router). The specified tunnel template must be configured using the `tunnel-end-point`	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	statement under the `[edit firewall]` or `[edit logical-systems *logical-system-name* firewall]` hierarchy level. For more information, see "Understanding Filter-Based Tunneling Across IPv4 Networks" on page 1394.	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`encapsulate` *`template-name`* (for L2TP tunnels)	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable filter-based L2TP tunneling using the specified tunnel template. You can configure a filter term that pairs this action with the appropriate match conditions, and then attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a configuration that attaches an encapsulating filter to an interface that does not support filter-based GRE tunneling, the system writes a syslog warning message that the interface does not support the filter.	• `family inet`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	When the interface receives a matched packet, processes that run on the Packet Forwarding Engine use information in the specified tunnel template to perform the following operations: 1. Attach an L2TP header (with or without a tunnel key value, as specified in the tunnel template). 2. Attach a header for the IPv4 transport protocol. 3. Forward the resulting L2TP packet from the tunnel source interface to the tunnel destination (the remote PE router). The specified tunnel template must be configured using the `tunnel-end-point` statement under the `[edit firewall]` or `[edit`	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	`logical-systems` *`logical-system-name`* `firewall]` statement hierarchy.	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`exclude-accounting`	Exclude the packet from being included in accurate accounting statistics for tunneled subscribers on an L2TP LAC. Typically used in filters that match DHCPv6 or ICMPv6 control traffic Failure to exclude these packets results in the idle-timeout detection mechanism considering these packets as data traffic, causing the timeout to never expire. (The idle timeout is configured with the `client-idle-timeout` and `client-idle-timeout-ingress-only` statements in the access profile session options.) The term excludes packets from being included in counts for both family accurate accounting and service accurate accounting. The packets are still included in the session interface statistics.	• `family inet` • `family inet6`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	The term is available for both `inet` and `inet6` families, but is used only for `inet6`.	
`logical-system` *`logical-system-name`*	Direct the packet to the specified logical system. **NOTE**: This action is not supported on PTX Series Packet Transport Routers.	• `family inet` • `family inet6`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`reject` *`message-type`*	Reject the packet and return an ICMPv4 or ICMPv6 message: • If no *`message-type`* is specified, a `destination unreachable` message is returned by default. • If `tcp-reset` is specified as the *`message-type`*, `tcp-reset` is returned only if the packet is a TCP packet. Otherwise, the `administratively-prohibited` message, which has a value of 13, is returned. • If any other *`message-type`* is specified, that message is returned. **NOTE**: Rejected packets can be sampled or logged if you configure the `sample` or `syslog` action. For MX2K-	• `family inet` • `family inet6`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	MPC11E, ICMP reject messages traverse egress filters, policers, and class of service (CoS) configurations and so are included in those statistics. The same is true for `destination unreachable` messages. The *`message-type`* can be one of the following values: `address-unreachable`, `administratively-prohibited`, `bad-host-tos`, `bad-network-tos`, `beyond-scope`, `fragmentation-needed`, `host-prohibited`, `host-unknown`, `host-unreachable`, `network-prohibited`, `network-unknown`, `network-unreachable`, `no-route`, `port-unreachable`, `precedence-cutoff`, `precedence-violation`, `protocol-unreachable`, `source-host-isolated`, `source-route-failed`, or `tcp-reset`. On PTX1000 routers, the reject	

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
	action is supported on ingress interfaces only.	
`routing-instance` *`instance-name`*	Direct the packet to the specified routing instance.	• `family inet` • `family inet6`

Table 39: Terminating Actions for Firewall Filters *(Continued)*

Terminating Action	Description	Protocols
`topology` *`topology-name`*	Direct the packet to the specified topology. **NOTE**: This action is not supported on PTX Series Packet Transport Routers. Each routing instance (primary or virtual-router) supports one default topology to which all forwarding classes are forwarded. For multitopology routing, you can configure a firewall filter on the ingress interface to match a specific forwarding class, such as expedited forwarding, with a specific topology. The traffic that matches the specified forwarding class is then added to the routing table for that topology.	• `family inet` • `family inet6`

NOTE: On QFX5120-48Y and QFX5120-32C switch models, configure `discard` action explicitly to bring down a BFD session. However, note that if there is a `port-mirror` action configured before the `discard` action, then the BFD session will not be brought down.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | 814
- Firewall Filter Nonterminating Actions | 871
- Firewall Filter Match Conditions for IPv4 Traffic | 940
- Firewall Filter Match Conditions for IPv6 Traffic | 957
- *enhanced-mode*
- Firewall Filter Flexible Match Conditions | 862
- *Firewall Filter Terminating and Nonterminating Actions for Protocol-Independent Traffic in Dynamic Service Profiles*

Firewall Filter Match Conditions and Actions (ACX Series Routers)

IN THIS SECTION

- Overview of Firewall Filter Match Conditions and Actions on ACX Series Routers | 910
- Match Conditions for Bridge Family Firewall Filters (ACX Series Routers) | 912
- Match Conditions for CCC Firewall Family Filters (ACX Series Routers) | 916
- Match Conditions for IPv4 Traffic (ACX Series Routers) | 917
- Match Conditions for IPv6 Traffic (ACX Series Routers) | 923
- Match Conditions for MPLS Traffic (ACX Series Routers) | 930
- Nonterminating Actions (ACX Series Routers) | 931
- Terminating Actions (ACX Series Routers) | 935

On ACX Series Universal Metro Routers, you can configure firewall filters to filter packets and to perform an action on packets that match the filter. The match conditions specified to filter the packets are specific to the type of traffic being filtered.

Firewall filters with IPv6 match conditions not supported at the `firewall family inet6 filter` *`name`* hierarchy level on ACX6360-OR routers in Junos OS Release 19.1R1.

NOTE: On ACX Series routers, the filter for the exiting traffic (egress filter) can be applied only for interface-specific instances of the *firewall filter*.

On ACX Series routers, TCAM errors are seen when you modify a prefix or a term on the applied firewall filters. To modify a prefix or a term in the firewall filter, you need to remove the existing firewall filter and then apply the modified filter.

NOTE: On ACX Series routers, you cannot apply a firewall filter in the egress direction on IRB interfaces.

Overview of Firewall Filter Match Conditions and Actions on ACX Series Routers

Table 40 on page 910 describes the types of traffic for which you can configure standard stateless firewall filters.

Table 40: Standard Firewall Filter Match Conditions by Protocol Family for ACX Series Routers

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
Protocol-independent	`[edit firewall family any filter` *`filter-name`* `term` *`term-name`*`]` No match conditions are supported for this traffic type on ACX Series routers.
IPv4	`[edit firewall family inet filter` *`filter-name`* `term` *`term-name`* For the complete list of match conditions, see "Match Conditions for IPv4 Traffic (ACX Series Routers)" on page 917.
MPLS	`[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`*`]` For the complete list of match conditions, see "Match Conditions for MPLS Traffic (ACX Series Routers)" on page 930.

Table 40: Standard Firewall Filter Match Conditions by Protocol Family for ACX Series Routers *(Continued)*

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
Layer 2 CCC	`[edit firewall family ccc filter` *`filter-name`* `term` *`term-name`*`]` No match conditions are supported for this traffic type on ACX Series routers.
Bridge	`[edit firewall family bridge filter filter-name term term-name]` `[edit firewall family ethernet-switching filter filter-name term term-name]` (Applicable to ACX5048 and ACX5096 routers only.)

On ACX5448 router, the following ingress family filters can be scaled based on the availability of external-tcam:

- family `ethernet-switching`
- family `ccc`
- family `inet`
- family `inet6`
- family `mpls`
- family `vpls`

Under the `then` statement for a standard stateless firewall filter term, you can specify the actions to be taken on a packet that matches the term.

Table 41 on page 912 summarizes the types of actions you can specify in a standard stateless firewall filter term.

Table 41: Standard Firewall Filter Action Categories for ACX Series Routers

Type of Action	Description	Comment
Terminating	Halts all evaluation of a firewall filter for a specific packet. The router performs the specified action, and no additional terms are used to examine the packet. You can specify only one *terminating action* in a standard firewall filter. You can, however, specify one terminating action with one or more *nonterminating actions* in a single term. For example, within a term, you can specify `accept` with `count` and `syslog`.	See "Terminating Actions (ACX Series Routers)" on page 935.
Nonterminating	Performs other functions on a packet (such as incriminating a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality), but any additional terms are used to examine the packet.	See "Nonterminating Actions (ACX Series Routers)" on page 931.

Match Conditions for Bridge Family Firewall Filters (ACX Series Routers)

IN THIS SECTION

- Bridge Family Firewall Filters on ACX Series Routers | **912**

Bridge Family Firewall Filters on ACX Series Routers

Bridge family firewall filters can be configured at the IFL-family level on ACX series routers. Bridge family filters are used to match the L2 bridge flows based on the supported Layer2/Layer3 fields and take firewall action. The maximum number of terms supported for bridge firewall filters on ACX Series routers is 124.

NOTE: On ACX5448 and ACX7000 series routers, you need to apply the layer 2 firewall filters only on the layer 2 switched packets, even if the bridge domain has IRB attached to the bridge domain. If the packet is layer 3 forwarded, then layer 3 filters must be applied on the IRB.

NOTE: On ACX Series routers, you cannot apply a firewall filter in the egress direction on IRB interfaces.

Table 42 on page 913 shows the match conditions supported for bridge family filters.

Table 42: Bridge Family Firewall Filter Match Conditions for ACX Series Routers

Match Condition	Description
apply-groups	Set the groups from which to inherit configuration data
apply-groups-except	Set which groups will not broadcast configuration data
destination-mac-address	Set the destination MAC address
destination-port	Match the TCP/UDP destination port
`destination-prefix-list`	Match IP destination prefixes in named list.
dscp	Match the Differentiated Services (DiffServ) code point
ether-type	Match the Ethernet type
icmp-code	Match a ICMP message code
icmp-type	Match a ICMP message type
interface-group	Match an interface group

Table 42: Bridge Family Firewall Filter Match Conditions for ACX Series Routers *(Continued)*

Match Condition	Description
ip-destination-address	Match an IP destination address
ip-precedence	Match an IP precedence value
ip-protocol	Match an IP protocol type
ip-source-address	Match an IP source address
learn-vlan-1p-priority	Match the learned 802.1p VLAN Priority
learn-vlan-dei	Match user VLAN ID DEI bit
learn-vlan-id	Match a learnt VLAN ID
source-mac-address	Set the source MAC address
`source-prefix-list`	Match IP source prefixes in named list.
source-port	Match a TCP/UDP source port
user-vlan-1p-priority	Match user 802.1p VLAN Priority
user-vlan-id	Match a user VLAN ID
vlan-ether-type	Match a VLAN Ethernet type

Table 43 on page 915 shows the action fields supported.

Table 43: Bridge Family Firewall Filter Action Fields for ACX Series Routers

Action Field	Description
accept	Accept the packet
count	Count the packet in the named counter
discard	Discard the packet
forwarding-class	Classify packet to forwarding class
loss-priority	Packet's loss priority
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 at the command-line interface (CLI).
policer	Name of policer to use to rate-limit traffic
syslog	Log the packet to the system log file.
three-color-policer	Police the packet using a three-colo-policer

NOTE: Bridge family firewall filters can be applied as an output filter on Layer 2 interfaces. When the Layer 2 interface is on a bridge-domain configured with the `vlan-id` statement, ACX series routers can match the outer-vlan of the packet using the user vlan-id match specified in the bridge family firewall filter.

Match Conditions for CCC Firewall Family Filters (ACX Series Routers)

IN THIS SECTION

- Match Conditions for CCC Family Firewall Filters | 916

Match Conditions for CCC Family Firewall Filters

On ACX Series routers, you can configure a standard firewall filter with match conditions for circuit cross-connection (CCC) traffic (family ccc).

Table 44 on page 916 describes the match conditions you can configure at the `[edit firewall family ccc filter filter-name term term-name]` hierarchy level.

Table 44: CCC Family Firewall Filter Match Conditions for ACX Series Routers

Field	Description
`destination-mac-address`	Destination MAC address
`destination-port`	Matches TCP/UDP destination port
`dscp`	Matches differentiated services (DiffServ) code point
`icmp-code`	Matches ICMP message code
`icmp-type`	Matches ICMP message type
`ip-destination-address`	Matches destination IP address
`ip-precedence`	Matches IP precedence value
`ip-protocol`	Matches IP protocol type

Table 44: CCC Family Firewall Filter Match Conditions for ACX Series Routers *(Continued)*

Field	Description
`ip-source-address`	Matches source IP address
`learn-vlan-1p-priority`	Matches learned 802.1p VLAN priority
`source-mac-address`	Source MAC address
`source-port`	Matches TCP/UDP source port
`user-vlan-1p-priority`	Matches user 802.1p VLAN priority

Match Conditions for IPv4 Traffic (ACX Series Routers)

On ACX Series routers, you can configure a standard stateless firewall filter with match conditions for IP version 4 (IPv4) traffic (`family inet`). Table 45 on page 917 describes the match conditions you can configure at the `[edit firewall family inet filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers

Match Condition	Description
`destination-address` *`address`*	Match the IPv4 destination address field. **NOTE**: On ACX Series routers, you can specify only one destination address. A list of IPv4 destination addresses is not supported.

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(Continued)*

Match Condition	Description
`destination-port` *number*	Match the UDP or TCP destination port field. If you configure this match condition, we recommend that you also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-prefix-list`	Match IP destination prefixes in named list.

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(Continued)*

Match Condition	Description

`dscp` *number*	Match the 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 *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.
	
	You can specify a numeric value from 0 through 63. To specify the value in hexadecimal form, include 0x as a prefix. To specify the value in binary form, include b as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46).
	
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	
	  • `af11` (10), `af12` (12), `af13` (14)
	
	  • `af21` (18), `af22` (20), `af23` (22)
	
	  • `af31` (26), `af32` (28), `af33` (30)
	
	  • `af41` (34), `af42` (36), `af43` (38)

`fragment-flags` *number*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): `dont-fragment` (0x4), `more-fragments` (0x2), or `reserved` (0x8).

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(Continued)*

Match Condition	Description
`icmp-code` *number*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `protocol icmp` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • 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)
`icmp-type` *number*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the `protocol icmp` match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `echo-reply` (0), `echo-request` (8), `info-reply` (16), `info-request` (15), `mask-request` (17), `mask-reply` (18), `parameter-problem` (12), `redirect` (5), `router-advertisement` (9), `router-solicit` (10), `source-quench` (4), `time-exceeded` (11), `timestamp` (13), `timestamp-reply` (14), or `unreachable` (3).

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(Continued)*

Match Condition	Description
`ip-options` *values*	Match the 8-bit IP option field, if present, to the specified value. ACX Series routers support only the `ip-options_any` match condition, which ensures that the packets are sent to the Packet Forwarding Engine for processing. **NOTE**: On ACX Series routers, you can specify only one IP option value. Configuring multiple values is not supported.
`precedence` *ip-precedence-field*	Match the 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.
`protocol` *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `no-next-header`, `ospf` (89), `pim` (103), `routing`, `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).
`source-address` *address*	Match the IPv4 address of the source node sending the packet.
`source-port` *number*	Match the UDP or TCP source port field. If you configure this match condition for IPv4 traffic, we recommend that you also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.
`source-prefix-list`	Match IP source prefixes in named list.

Table 45: Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(Continued)*

Match Condition	Description
`tcp-flags` *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the `tcp-initial` match conditions. If you configure this match condition, we recommend that you also configure the `protocol tcp` match statement in the same term to specify that the TCP protocol is being used on the port.
`tcp-initial`	Match the initial packet of a TCP connection. This is an alias for `tcp-flags "(!ack & syn)"`. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the `protocol tcp` match condition in the same term.
`ttl` *number*	Match the IPv4 time-to-live number. Specify a TTL value or a range of TTL values. For *number*, you can specify one or more values from 2 through 255.

Match Conditions for IPv6 Traffic (ACX Series Routers)

You can configure a firewall filter with match conditions for Internet Protocol version 6 (IPv6) traffic (`family inet6`). Table 46 on page 923 describes the match conditions you can configure at the `[edit firewall family inet6 filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic

Match Condition	Description
`destination-address` *`address`*	Match the IPv6 destination address field.
`destination-port` *`number`*	Match the UDP or TCP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-prefix-list`	Match IP destination prefixes in named list.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`extension-headers` *`header-type`*	Match an extension header type that is contained in the packet by identifying a Next Header value. In the first fragment of a packet, the filter searches for a match in any of the extension header types. When a packet with a fragment header is found (a subsequent fragment), the filter only searches for a match of the next extension header type because the location of other extension headers is unpredictable. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `destination` (60), `esp` (50), `fragment` (44), `hop-by-hop` (0), `mobility` (135), or `routing` (43). To match *any* value for the extension header option, use the text synonym `any`. **NOTE**: Only the first extension header of the IPv6 packet can be matched. L4 header beyond one IPv6 extension header will be matched.
`hop-limit` *`hop-limit`*	Match the hop limit to the specified hop limit or set of hop limits. For *`hop-limit`*, specify a single value or a range of values from 0 through 255.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`icmp-code` *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `administratively-prohibited` (1), `address-unreachable` (3), `no-route-to-destination` (0), `port-unreachable` (4)

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`icmp-type` *`message-type`*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `certificate-path-advertisement` (149), `certificate-path-solicitation` (148), `destination-unreachable` (1), `echo-reply` (129), `echo-request` (128), `home-agent-address-discovery-reply` (145), `home-agent-address-discovery-request` (144), `inverse-neighbor-discovery-advertisement` (142), `inverse-neighbor-discovery-solicitation` (141), `membership-query` (130), `membership-report` (131), `membership-termination` (132), `mobile-prefix-advertisement-reply` (147), `mobile-prefix-solicitation` (146), `neighbor-advertisement` (136), `neighbor-solicit` (135), `node-information-reply` (140), `node-information-request` (139), `packet-too-big` (2), `parameter-problem` (4), `private-experimentation-100` (100), `private-experimentation-101` (101), `private-experimentation-200` (200), `private-experimentation-201` (201), `redirect` (137), `router-advertisement` (134), `router-renumbering` (138), `router-solicit` (133), or `time-exceeded` (3). For `private-experimentation-201` (201), you can also specify a range of values within square brackets.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`next-header` *header-type*	Match the first 8-bit Next Header field in the packet. Support for the `next-header` firewall match condition is available in Junos OS Release 13.3R6 and later. For IPv6, we recommend that you use the `payload-protocol` term rather than the `next-header` term when configuring a firewall filter with match conditions. Although either can be used, `payload-protocol` provides the more reliable match condition because it uses the actual payload protocol to find a match, whereas `next-header` simply takes whatever appears in the first header following the IPv6 header, which may or may not be the actual protocol. In addition, if `next-header` is used with IPv6, the accelerated filter block lookup process is bypassed and the standard filter used instead. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstops` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `mobility` (135), `no-next-header` (59), `ospf` (89), `pim` (103), `routing` (43), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112). **NOTE**: `next-header icmp6` and `next-header icmpv6` match conditions perform the same function. `next-header icmp6` is the preferred option. `next-header icmpv6` is hidden in the Junos OS CLI.
`source-address` *address*	Match the IPv6 address of the source node sending the packet.
`source-port` *number*	Match the UDP or TCP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.
`source-prefix-list`	Match IP source prefixes in named list.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`tcp-flags` *`flags`*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the `tcp-established` and `tcp-initial` match conditions. If you configure this match condition, we recommend that you also configure the `next-header tcp` match condition in the same term to specify that the TCP protocol is being used on the port.
`tcp-initial`	Match the initial packet of a TCP connection. This is a text synonym for `tcp-flags "(! ack & syn)"`. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the `next-header tcp` match condition in the same term.

Table 46: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description

`traffic-class` *number*	Match the 8-bit field that specifies the class-of-service (CoS) priority of the packet.
	
	This field was previously used as the type-of-service (ToS) field in IPv4.
	
	You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46).
	
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	
	  • `af11` (10), `af12` (12), `af13` (14)
	
	  • `af21` (18), `af22` (20), `af23` (22)
	
	  • `af31` (26), `af32` (28), `af33` (30)
	
	  • `af41` (34), `af42` (36), `af43` (38)

NOTE: If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see IPv6 Overview and Supported IPv6 Standards.

The following is a sample firewall family inet6 configuration:

```
user@host# show firewall family inet6
filter ipv6-filter {
    term t1 {
        from {
            source-address {
                2001:0000:0020:0020:0000:0000:0000:0150/128;
```

```
            }
            destination-address {
                2001:0000:0040:0040:0000:0000:0000:0150/128;
            }
            next-header tcp;
            source-port 1000;
            destination-port 2000;
            extension-header dstopts;
            traffic-class ef;
            tcp-flags 0x20;
            hop-limit 254;
        }
        then count ipv6-t1-count;
    }
    term t2 {
        from {
            icmp-type neighbor-solicit;
        }
        then count ipv6-t2-count;
    }
}
```

Match Conditions for MPLS Traffic (ACX Series Routers)

On ACX Series routers, you can configure a standard stateless firewall filter with match conditions for MPLS traffic (`family mpls`).

NOTE: The `input-list` *`filter-names`* and `output-list` *`filter-names`* statements for firewall filters for the `mpls` protocol family are supported on all interfaces with the exception of management interfaces and internal Ethernet interfaces (`fxp` or `em0`), loopback interfaces (`lo0`), and USB modem interfaces (`umd`).

Table 47 on page 931 describes the match conditions you can configure at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 47: Standard Firewall Filter Match Conditions for MPLS Traffic on ACX Series Routers

Match Condition	Description
`exp` *number*	Experimental (EXP) bit number or range of bit numbers in the MPLS header. For *number*, you can specify one or more values from 0 through 7 in decimal, binary, or hexadecimal format.

Nonterminating Actions (ACX Series Routers)

Standard stateless firewall filters support different sets of nonterminating actions for each protocol family.

NOTE: ACX Series routers do not support the `next term` action.

ACX Series routers support log and syslog actions in ingress and egress directions for family **inet** and family **bridge**.

ACX5448, ACX710 and ACX7100 series routers do not support **log**, **syslog**, **reject**, **forwarding-class**, and **loss-priority** in the egress direction. In the ingress and egress direction, the routers support interface specific semantics only.

Table 48 on page 931 describes the nonterminating actions you can configure for a standard firewall filter term.

Table 48: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers

Nonterminating Action	Description	Protocol Families
`count` *counter-name*	Count the packet in the named counter.	• `family any` • `family inet` • • `family mpls` • `family ccc` • `family bridge` • `family vpls`

Table 48: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(Continued)*

Nonterminating Action	Description	Protocol Families
`forwarding-class` *`class-name`*	Classify the packet based on the specified forwarding class: • `assured-forwarding` • `best-effort` • `expedited-forwarding` • `network-control` **NOTE**: This action is supported on ingress only.	• `family inet` • `family inet6` • • `family mpls` • `family ccc` • `family bridge` • `family vpls`
`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 at the command-line interface (CLI). **NOTE**: This action is supported on ingress and egress. The action on egress is not supported for family inet6.	• `family inet` • `family inet6` • `family bridge`

Table 48: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(Continued)*

Nonterminating Action	Description	Protocol Families
`loss-priority (high | medium-high | low)`	Set the packet loss priority (PLP) level. You cannot also configure the `three-color-policer` nonterminating action for the same firewall filter term. These two nonterminating actions are mutually exclusive. You must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can configure only the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*. **NOTE**: This action is supported on ingress only.	• `family any` • `family inet` • `family inet6` • `family mpls` • `family ccc` • `family bridge` • `family vpls`

Table 48: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(Continued)*

Nonterminating Action	Description	Protocol Families
`policer` *`policer-name`*	Name of policer to use to rate-limit traffic.	• `family any` • `family inet` • `family inet6` • `family mpls` • `family ccc` • `family bridge` • `family vpls`
`port-mirror`	Port-mirror the packet based on the specified family. **NOTE**: This action is supported on ingress only. ACX5048 and ACX5096 routers do not support **port-mirror**.	`family inet`
`syslog`	Log the packet to the system log file. **NOTE**: This action is supported on ingress and egress. The action on egress is not supported for family inet6.	• `family inet` • `family inet6` • `family bridge`

Table 48: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(Continued)*

Nonterminating Action	Description	Protocol Families
`three-color-policer (single-rate | two-rate)` *policer-name*	Police the packet using the specified single-rate or two-rate three-color policer. You cannot also configure the `loss-priority` action for the same firewall filter term. These two actions are mutually exclusive.	• `family any` • `family inet` • `family inet6` • `family mpls` • `family ccc` • `family bridge` • `family vpls`
traffic-class	Set traffic-class code point **NOTE**: This action is supported on ingress only.	`family inet6`

Terminating Actions (ACX Series Routers)

Standard stateless firewall filters support different sets of terminating actions for each protocol family.

NOTE: ACX Series routers do not support the `next term` action.

Table 49 on page 936 describes the terminating actions you can specify in a standard firewall filter term.

Table 49: Terminating Actions for Standard Firewall Filters on ACX Series Routers

Terminating Action	Description	Protocols
`accept`	Accept the packet.	• `family any` • `family inet` • `family mpls` • `family ccc`
`discard`	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are available for logging and sampling.	• `family any` • `family inet` • `family mpls` • `family ccc`

Table 49: Terminating Actions for Standard Firewall Filters on ACX Series Routers *(Continued)*

Terminating Action	Description	Protocols
`reject` *`message-type`*	Reject the packet and return an ICMPv4 or ICMPv6 message: • If no message type is specified, a `destination-unreachable` message is returned by default. • If `tcp-reset` is specified as the message type, `tcp-reset` is returned only if the packet is a TCP packet. Otherwise, the `administratively-prohibited` message, which has a value of 13, is returned. • If any other message type is specified, that message is returned. **NOTE**: • Rejected packets can be sampled or logged if you configure the `sample` or `syslog` action. • This action is supported on ingress only. The *`message-type`* option can have one of the following values: `address-unreachable`, `administratively-prohibited`, `bad-host-tos`, `bad-network-tos`, `beyond-scope`, `fragmentation-needed`, `host-prohibited`, `host-unknown`, `host-unreachable`, `network-prohibited`, `network-unknown`, `network-unreachable`, `no-route`, `port-unreachable`, `precedence-cutoff`, `precedence-violation`, `protocol-unreachable`, `source-host-isolated`, `source-route-failed`, or `tcp-reset`.	`family inet`
`routing-instance` *`routing-instance-name`*	Direct the packet to the specified routing instance.	• `family inet`

Firewall Filter Match Conditions for Protocol-Independent Traffic

You can configure a firewall filter with match conditions for protocol-independent traffic (`family any`).

To apply a protocol-independent firewall filter to a logical interface, configure the `filter` statement under the logical unit.

NOTE: On MX Series routers, attach a protocol-independent firewall filter to a logical interface by configuring the `filter` statement *directly* under the logical unit:

- `[edit interfaces name unit number filter]`
- `[edit logical-systems name interfaces name unit number filter]`

On all other supported devices, attach a protocol-independent firewall filter to a logical interface by configuring the `filter` statement under the protocol family (`family any`):

- `[edit interfaces name unit number family any filter]`
- `[edit logical-systems name interfaces name unit number family any filter]`

Table 50 on page 938 describes the *match-conditions* you can configure at the `[edit firewall family any filter filter-name term term-name from]` hierarchy level.

Table 50: Firewall Filter Match Conditions for Protocol-Independent Traffic

Match Condition	Description
`forwarding-class class`	Match the forwarding class of the packet. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`. For information about forwarding classes and router-internal output queues, see *Understanding How Forwarding Classes Assign Classes to Output Queues*. **NOTE**: On T4000 Type 5 FPCs, a filter attached at the Layer 2 application point (that is, at the logical interface level) is unable to match with the forwarding class of a packet that is set by a Layer 3 classifier such as DSCP, DSCP V6, `inet-precedence`, and `mpls-exp`.
`forwarding-class-except class`	Do not match on the forwarding class. For details, see the `forwarding-class` match condition.
`interface interface-name`	Match the interface on which the packet was received. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.

Table 50: Firewall Filter Match Conditions for Protocol-Independent Traffic *(Continued)*

Match Condition	Description
`interface-set` *`interface-set-name`*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.
`loss-priority` *`level`*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. **NOTE**: This match condition is not supported on PTX series packet transport routers. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.
`loss-priority-except` *`level`*	Do not match the PLP level. For details, see the `loss-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers.
`packet-length` *`bytes`*	Match the 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. You can also specify a range of values to be matched.
`packet-length-except` *`bytes`*	Do not match on the received packet length, in bytes. For details, see the `packet-length` match type.

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters | 814

Firewall Filter Terminating Actions | 884

Firewall Filter Nonterminating Actions | 871

Firewall Filter Match Conditions for IPv4 Traffic

You can configure a firewall filter with match conditions for Internet Protocol version 4 (IPv4) traffic (`family inet`).

NOTE: For MX Series routers with MPCs, you need to initialize the filter counter for Trio-only match filters in the MIB by walking the corresponding SNMP MIB, for example, `show snmp mib walk` *`name`* `ascii`. This forces Junos to learn the filter counters, and ensures that the filter statistics are displayed (this is because the first poll to filter statistics may not show all counters). This guidance applies to all enhanced mode firewall filters, filters with flexible conditions, and filters with certain terminating actions. See those topics, listed under Related Documentation, for details.

Table 51 on page 940 describes the *`match-conditions`* you can configure at the `[edit firewall family inet filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic

Match Condition	Description
`address` *`address`* `[ except ]`	Match the IPv4 source or destination address field unless the `except` option is included. If the option is included, do not match the IPv4 source or destination address field. The `except` modifier is not supported on EX2300 and EX3400 platforms.
`ah-spi` *`spi-value`*	(M Series routers, except M120 and M320) Match the IPsec authentication header (AH) security parameter index (SPI) value. **NOTE**: This match condition is not supported on PTX series routers.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`ah-spi-except` *`spi-value`*	(M Series routers, except M120 and M320) Do not match the IPsec AH SPI value. **NOTE**: This match condition is not supported on PTX series routers.
`apply-groups`	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.
`destination-address` *`address`* `[ except ]`	Match the IPv4 destination address field unless the `except` option is included. If the option is included, do not match the IPv4 destination address field. You cannot specify both the `address` and `destination-address` match conditions in the same term.
`destination-class` *`class-names`*	Match one or more specified destination class names (sets of destination prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 992.
`destination-class-except` *`class-names`*	Do not match one or more specified destination class names. For details, see the `destination-class` match condition.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`destination-port` *number*	Match the UDP or TCP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term. When configuring port based matches you must also configure the `protocol udp` or `protocol tcp` match statement in the same filter term. Matching only on the port value can result in unexpected matches. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-port-except` *number*	Do not match the UDP or TCP destination port field. For details, see the `destination-port` match condition.
`destination-prefix-list` *name* `[ except ]`	Match destination prefixes in the specified list unless the `except` option is included. If the option is included, do not match the destination prefixes in the specified list. Specify the name of a prefix list defined at the `[edit policy-options prefix-list` *prefix-list-name*] hierarchy level.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description

`dscp` *number*	Match the 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 *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.
	
	Support was added for filtering on Differentiated Services Code Point (DSCP) and forwarding class for Routing Engine sourced packets, including IS-IS packets encapsulated in generic routing encapsulation (GRE). Subsequently, when upgrading from a previous version of Junos OS where you have both a class of service (CoS) and firewall filter, and both include DSCP or forwarding class filter actions, the criteria in the firewall filter automatically takes precedence over the CoS settings. The same is true when creating new configurations; that is, where the same settings exist, the firewall filter takes precedence over the CoS, regardless of which was created first.
	
	You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46).
	
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	
	  • `af11` (10), `af12` (12), `af13` (14)
	
	  • `af21` (18), `af22` (20), `af23` (22)
	
	  • `af31` (26), `af32` (28), `af33` (30)
	
	  • `af41` (34), `af42` (36), `af43` (38)

`dscp-except` *number*	Do not match on the DSCP number. For more information, see the `dscp` match condition.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description	
`esp-spi` *spi-value*	Match the 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. **NOTE**: This match condition is not supported on PTX series routers.	
`esp-spi-except` *spi-value*	Match the IPsec ESP SPI value. Do not match on this specific SPI value. **NOTE**: This match condition is not supported on PTX series routers.	
`first-fragment`	Match if the packet is the first fragment of a fragmented packet. Do not match if the packet is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of `0`. This match condition is an alias for the bit-field match condition `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: `first-fragment` and `is-fragment`.	
`flexible-match-mask` *value*	`bit-length`	Length of the data to be matched in bits, not needed for string input (0..128)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-mask-name`	Select a flexible match from predefined template field
	`mask-in-hex`	Mask out bits in the packet data to be matched
	`match-start`	Start point to match in packet

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description	
	`prefix`	Value data/string to be matched
`flexible-match-range` *value*	`bit-length`	Length of the data to be matched in bits (0..32)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-range-name`	Select a flexible match from predefined template field
	`match-start`	Start point to match in packet
	`range`	Range of values to be matched
	`range-except`	Do not match this range of values
`forwarding-class` *class*	Match the forwarding class of the packet. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`. For information about forwarding classes and router-internal output queues, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.	
`forwarding-class-except` *class*	Do not match the forwarding class of the packet. For details, see the `forwarding-class` match condition.	

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`fragment-flags` *number*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): `dont-fragment` (0x4), `more-fragments` (0x2), or `reserved` (0x8).
`fragment-offset` *value*	Match the 13-bit fragment offset field in the IP header. The value is the offset, in 8-byte units, in the overall datagram message to the data fragment. Specify a numeric value, a range of values, or a set of values. An offset value of `0` indicates the first fragment of a fragmented packet. The `first-fragment` match condition is an alias for the `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`).
`fragment-offset-except` *number*	Do not match the 13-bit fragment offset field.
`gre-key` *range*	Match the gre-key field. The GRE key field is a 4 octet number inserted by the GRE encapsulator. It is an optional field for use in GRE encapsulation. The *range* can be a single GRE key number or a range of key numbers. For MX Series routers with MPCs, initialize new firewall filters that include this condition by walking the corresponding SNMP MIB.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description

`icmp-code` *number*	Match the ICMP message code field.
	
	NOTE: When using this match condition, you should also use the `protocol icmp` match condition in the same term (as shown below) to ensure that **icmp** packets are being evaluated.
	
	`term Allow _ICMP {`
	`from protocol icmp {`
	`icmp-code ip-header-bad;`
	`icmp-type echo-reply;`
	`}`
	`then {`
	`policer ICMP_Policier;`
	`count Allow_ICMP;`
	
	You must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type.
	
	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:
	
	• 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)

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`icmp-code-except` *`message-code`*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.

`icmp-type` *`number`*	Match the ICMP message type field.
	
	NOTE: When using this match condition, you should also use the `protocol icmp` match condition in the same term (as shown below) to ensure that **icmp** packets are being evaluated.
	
	`term Allow _ICMP {`
	`from protocol icmp {`
	`icmp-code ip-header-bad;`
	`icmp-type echo-reply;`
	`}`
	`then {`
	`policer ICMP_Policier;`
	`count Allow_ICMP;`
	
	You must also configure the `icmp-type` *`message-type`* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type.
	
	NOTE: For Junos OS Evolved, you must configure the `protocol` match statement in the same term.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `echo-reply` (0), `echo-request` (8), `info-reply` (16), `info-request` (15), `mask-request` (17), `mask-reply` (18), `parameter-problem` (12), `redirect` (5), `router-advertisement` (9), `router-solicit` (10), `source-quench` (4), `time-exceeded` (11), `timestamp` (13), `timestamp-reply` (14), or `unreachable` (3).

`icmp-type-except` *`message-type`*	Do not match the ICMP message type field. For details, see the `icmp-type` match condition.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`interface` *interface-name*	Match the interface on which the packet was received. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.
`interface-group` *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from 0 through 255. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the `[interfaces` *interface-name* `unit` *number* `family` *family* `filter group]` hierarchy level. **NOTE**: This match condition is not supported on PTX series routers. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.
`interface-group-except` *group-number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the `interface-group` match condition. **NOTE**: This match condition is not supported on PTX series routers.
`interface-set` *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. **NOTE**: This match condition is not supported on PTX series routers. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`ip-options` *values*	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): `loose-source-route` (131), `record-route` (7), `router-alert` (148), `security` (130), `stream-id` (136),`strict-source-route` (137), or `timestamp` (68). To match *any* value for the IP option, use the text synonym `any`. To match on *multiple* values, specify the list of values within square brackets ('[' and ']'). To match a *range* of values, use the value specification `[` *value1-value2* `]`. For example, the match condition `ip-options [ 0-147 ]` matches on an IP options field that contains the `loose-source-route`, `record-route`, or `security` values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the `router-alert` value (148). For most interfaces, a filter term that specifies an `ip-option` match on one or more *specific* IP option values (a value other than `any`) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an `ip-option` match on one or more specific IP option values, you cannot specify the `count`, `log`, or `syslog` nonterminating actions *unless* you also specify the `discard` terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router. • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the `ip-options any` match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 100-Gigabit Ethernet MPC, 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers are capable of parsing the IP option field of the IPv4 packet header. For interfaces configured on those MPCs, *all* packets that are matched using the `ip-options` match condition are sent to the Packet Forwarding Engine for processing. The `ip-options` *any* match condition is supported on PTX10003 and PTX10008 Series routers starting with Junos Evolved OS Release 20.2R1. **NOTE**:

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
	• On MX series routers, filter matches using `ip-options` cannot be used with egress (output) filters. • On M and T series routers, firewall filters cannot count `ip-options` packets on a per option type and per interface basis. A limited work around is to use the `show pfe statistics ip options` command to see `ip-options` statistics on a per PFE basis. See show pfe statistics ip for sample output.
`ip-options-except` *`values`*	Do not match the IP option field to the specified value or list of values. For details about specifying the *`values`*, see the `ip-options` match condition.
`is-fragment`	Using this condition causes a match if the More Fragments flag is enabled in the IP header and if the fragment offset is not zero. **NOTE**: To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`).
`loss-priority` *`level`*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`loss-priority-except` *level*	Do not match the PLP level. For details, see the `loss-priority` match condition.
`packet-length` *bytes*	Match the 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. You can also specify a range of values to be matched.
`packet-length-except` *bytes*	Do not match the length of the received packet, in bytes. For details, see the `packet-length` match type.
`port` *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the `destination-port` match condition or the `source-port` match condition in the same term. When configuring port based matches you must also configure the `protocol udp` or `protocol tcp` match statement in the same filter term. Matching only on the port value can result in unexpected matches. In place of the numeric value, you can specify one of the text synonyms listed under `destination-port`.
`port-except` *number*	Do not match either the source or destination UDP or TCP port field. For details, see the `port` match condition.
`precedence` *ip-precedence-value*	Match the 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.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`precedence-except` *ip-precedence-value*	Do not match the 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.
`prefix-list` *name* `[ except ]`	Match the prefixes of the source or destination address fields to the prefixes in the specified list unless the `except` option is included. If the option is included, do not match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the `[edit policy-options prefix-list` *prefix-list-name*`]` hierarchy level. **NOTE**: This match condition is not supported on PTX1000 routers.
`protocol` *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).
`protocol-except` *number*	Do not match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description

`rat-type` *tech-type-value*	Match the radio-access technology (RAT) type specified in the 8-bit Tech-Type field of Proxy Mobile IPv4 (PMIPv4) access technology type extension. The technology type specifies the access technology through which the mobile device is connected to the access network.
	
	Specify a single value, a range of values, or a set of values. You can specify a technology type as a numeric value from 0 through 255 or as a system keyword.
	
	• The following numeric values are examples of well-known technology types:
	  • Numeric value 1 matches IEEE 802.3.
	  • Numeric value 2 matches IEEE 802.11a/b/g.
	  • Numeric value 3 matches IEEE 802.16e
	  • Numeric value 4 matches IEEE 802.16m.
	• Text string `eutran` matches 4G.
	• Text string `geran` matches 2G.
	• Text string `utran` matches 3G.

`rat-type-except` *tech-type-value*	Do not match the RAT Type.
`service-filter-hit`	Match a packet received from a filter where a `service-filter-hit` action was applied. **NOTE**: This match condition is not supported on PTX series routers.
`source-address` *address* `[ except ]`	Match the IPv4 address of the source node sending the packet unless the `except` option is included. If the option is included, do not match the IPv4 address of the source node sending the packet. You cannot specify both the `address` and `source-address` match conditions in the same term.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`source-class` *class-names*	Match one or more specified source class names (sets of source prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 992.
`source-class-except` *class-names*	Do not match one or more specified source class names. For details, see the `source-class` match condition.
`source-port` *number*	Match the UDP or TCP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. When configuring port based matches you must also configure the `protocol udp` or `protocol tcp` match statement in the same filter term. Matching only on the port value can result in unexpected matches. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.
`source-port-except` *number*	Do not match the UDP or TCP source port field. For details, see the `source-port` match condition.
`source-prefix-list` *name* `[ except ]`	Match source prefixes in the specified list unless the `except` option is included. If the option is included, do not match the source prefixes in the specified list. Specify the name of a prefix list defined at the `[edit policy-options prefix-list` *prefix-list-name*] hierarchy level.
`tcp-established`	Match TCP packets of an established TCP session (packets other than the first packet of a connection). This is an alias for `tcp-flags "(ack | rst)"`. This match condition does not implicitly check that the protocol is TCP. To check this, specify the `protocol tcp` match condition.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`tcp-flags` *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the `tcp-established` and `tcp-initial` match conditions. If you configure this match condition, we recommend that you also configure the `protocol tcp` match statement in the same term to specify that the TCP protocol is being used on the port. For IPv4 traffic only, this match condition does not implicitly check whether the datagram contains the first fragment of a fragmented packet. To check for this condition for IPv4 traffic only, use the `first-fragment` match condition.
`tcp-initial`	Match the initial packet of a TCP connection. This is an alias for `tcp-flags "(!ack & syn)"`. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the `protocol tcp` match condition in the same term.

Table 51: Firewall Filter Match Conditions for IPv4 Traffic *(Continued)*

Match Condition	Description
`ttl` *number*	Match the IPv4 time-to-live number. Specify a TTL value or a range of TTL values. For *number*, 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` *number*	Do not match on the IPv4 TTL number. For details, see the `ttl` match condition.

Release History Table

Release	Description
13.3R7	Support was added for filtering on Differentiated Services Code Point (DSCP) and forwarding class for Routing Engine sourced packets, including IS-IS packets encapsulated in generic routing encapsulation (GRE).

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters | **814**

Firewall Filter Terminating Actions | **884**

Firewall Filter Nonterminating Actions | **871**

Firewall Filter Match Conditions for IPv6 Traffic | **957**

enhanced-mode

Firewall Filter Flexible Match Conditions | **862**

Firewall Filter Match Conditions for IPv6 Traffic

You can configure a firewall filter with match conditions for Internet Protocol version 6 (IPv6) traffic (`family inet6`).

NOTE: For MX Series routers with MPCs, you need to initialize the filter counter for Trio-only match filters by walking the corresponding SNMP MIB, for example, `show snmp mib walk` *name* `ascii`.

This forces Junos to learn the filter counters and ensure that the filter statistics are displayed. This guidance applies to all enhanced mode firewall filters, filters with flexible conditions, and filters with the certain terminating actions. See those topics, listed under Related Documentation, for details.

Table 52 on page 958 describes the match conditions you can configure at the `[edit firewall family inet6 filter filter-name term term-name from]` hierarchy level.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic

Match Condition	Description
`address address [ except ]`	Match the IPv6 source or destination address field unless the `except` option is included. If the option is included, do not match the IPv6 source or destination address field.
`apply-groups`	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.
`destination-address address [ except ]`	Match the IPv6 destination address field unless the `except` option is included. If the option is included, do not match the IPv6 destination address field. You cannot specify both the `address` and `destination-address` match conditions in the same term.
`destination-class class-names`	Match one or more specified destination class names (sets of destination prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 992.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`destination-class-except` *`class-names`*	Do not match one or more specified destination class names. For details, see the `destination-class` match condition.
`destination-port` *`number`*	Match the UDP or TCP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. **NOTE**: For Junos OS Evolved, you must configure the `next-header` match statement in the same term. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-port-except` *`number`*	Do not match the UDP or TCP destination port field. For details, see the `destination-port` match condition.
`destination-prefix-list` *`prefix-list-name`* `[ except ]`	Match the IPv6 destination prefix to the specified list unless the `except` option is included. If the option is included, do not match the IPv6 destination prefix to the specified list. The prefix list is defined at the [`edit policy-options prefix-list` *`prefix-list-name`*] hierarchy level.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description	
`extension-headers` *`header-type`*	Match an extension header type that is contained in the packet by identifying a Next Header value. **NOTE**: This match condition is only supported on MPCs in MX Series routers. In the first fragment of a packet, the filter searches for a match in any of the extension header types. When a packet with a fragment header is found (a subsequent fragment), the filter only searches for a match of the next extension header type because the location of other extension headers is unpredictable. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `destination` (60), `esp` (50), `fragment` (44), `hop-by-hop` (0), `mobility` (135), or `routing` (43). To match *any* value for the extension header option, use the text synonym `any`. For MX Series routers with MPCs, initialize new firewall filters that include this condition by walking the corresponding SNMP MIB.	
`first-fragment`	Match if the packet is the first fragment.	
`extension-headers-except` *`header-type`*	Do not match an extension header type that is contained in the packet. For details, see the `extension-headers` match condition. **NOTE**: This match condition is only supported on MPCs in MX Series routers.	
`flexible-match-mask` *`value`*	`bit-length`	Length of integer input (1..32 bits); (Optional) Length of string input (1..128 bits)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description	
	`flexible-mask-name`	Select a flexible match from predefined template field
	`mask-in-hex`	Mask out bits in the packet data to be matched
	`match-start`	Start point to match in packet
	`prefix`	Value data/string to be matched
	See "Firewall Filter Flexible Match Conditions" on page 862 for details	
`flexible-match-range` *value* Ranges should use the following format: *Integer-Integer*	`bit-length`	Length of the data to be matched in bits (0..32)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-range-name`	Select a flexible match from predefined template field
	`match-start`	Start point to match in packet
	`range`	Range of values to be matched
	`range-except`	Do not match this range of values
	See "Firewall Filter Flexible Match Conditions" on page 862 for details	

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`forwarding-class` *class*	Match the forwarding class of the packet. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`. For information about forwarding classes and router-internal output queues, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.
`forwarding-class-except` *class*	Do not match the forwarding class of the packet. For details, see the `forwarding-class` match condition.
`hop-limit` *hop-limit*	Match the hop limit to the specified hop limit or set of hop limits. For *hop-limit*, specify a single value or a range of values from 0 through 255. Supported on interfaces hosted on MICs or MPCs in MX Series routers only. **NOTE**: This match condition is supported on PTX series routers when `enhanced-mode` is configured on the router.
`hop-limit-except` *hop-limit*	Do not match the hop limit to the specified hop limit or set of hop limits. For details, see the `hop-limit` match condition. Supported on interfaces hosted on MICs or MPCs in MX Series routers only. **NOTE**: This match condition is supported on PTX series routers when `enhanced-mode` is configured on the router.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`icmp-code` *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `administratively-prohibited` (1), `address-unreachable` (3), `no-route-to-destination` (0), `port-unreachable` (4)
`icmp-code-except` *message-code*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`icmp-type` *message-type*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. **NOTE**: For Junos OS Evolved, you must configure the `next-header` match statement in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `certificate-path-advertisement` (149), `certificate-path-solicitation` (148), `destination-unreachable` (1), `echo-reply` (129), `echo-request` (128), `home-agent-address-discovery-reply` (145), `home-agent-address-discovery-request` (144), `inverse-neighbor-discovery-advertisement` (142), `inverse-neighbor-discovery-solicitation` (141), `membership-query` (130), `membership-report` (131), `membership-termination` (132), `mobile-prefix-advertisement-reply` (147), `mobile-prefix-solicitation` (146), `neighbor-advertisement` (136), `neighbor-solicit` (135), `node-information-reply` (140), `node-information-request` (139), `packet-too-big` (2), `parameter-problem` (4), `private-experimentation-100` (100), `private-experimentation-101` (101), `private-experimentation-200` (200), `private-experimentation-201` (201), `redirect` (137), `router-advertisement` (134), `router-renumbering` (138), `router-solicit` (133), or `time-exceeded` (3). For `private-experimentation-201` (201), you can also specify a range of values within square brackets.
`icmp-type-except` *message-type*	Do not match the ICMP message type field. For details, see the `icmp-type` match condition.
`interface` *interface-name*	Match the interface on which the packet was received. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`interface-group` *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from `0` through `255`. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the `[interfaces` *interface-name* `unit` *number* `family` *family* `filter group]` hierarchy level. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.
`interface-group-except` *group-number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the `interface-group` match condition.
`interface-set` *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`ip-options` *values*	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): `loose-source-route` (131), `record-route` (7), `router-alert` (148), `security` (130), `stream-id` (136),`strict-source-route` (137), or `timestamp` (68). To match *any* value for the IP option, use the text synonym `any`. To match on *multiple* values, specify the list of values within square brackets ('[' and ']'). To match a *range* of values, use the value specification `[` *value1-value2* `]`. For example, the match condition `ip-options [ 0-147 ]` matches on an IP options field that contains the `loose-source-route`, `record-route`, or `security` values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the `router-alert` value (148). For most interfaces, a filter term that specifies an `ip-option` match on one or more *specific* IP option values (a value other than `any`) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an `ip-option` match on one or more specific IP option values, you cannot specify the `count`, `log`, or `syslog` nonterminating actions *unless* you also specify the `discard` terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router. • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the `ip-options any` match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 100-Gigabit Ethernet MPC, 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers are capable of parsing the IP option field of the IPv4 packet header. For interfaces configured on those MPCs, *all* packets that are matched using the `ip-options` match condition are sent to the Packet Forwarding Engine for processing.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description	
`ip-options-except` *values*	Do not match the IP option field to the specified value or list of values. For details about specifying the *values*, see the `ip-options` match condition.	
`is-fragment`	Match if the packet is a fragment.	
`last-fragment`	Match if the packet is the last fragment.	
`loss-priority` *level*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, T Series routers and EX Series switches with Enhanced II Flexible PIC Concentrators (FPCs), you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.	
`loss-priority-except` *level*	Do not match the PLP level. For details, see the `loss-priority` match condition.	

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`next-header` *`header-type`*	Match the first 8-bit Next Header field in the packet. Support for the `next-header` firewall match condition is available in Junos OS Release 13.3R6 and later. **NOTE**: MX platforms have a `next-header` match that matches the first Next Header (NH) in the packet and the `payload-protocol` to match the last NH. Whereas EVO-PTX platforms supported the `next-header` match on the last NH, but not the first one. The most common use case is to match the last NH, and this was native to the PTX platforms. Now the `next-header` matches the first NH, and the `payload-protocol` matches the last NH, and behaves the same way as on MX platforms. If you are using the IPv6 filter `next-header` clause on your WAN interfaces' firewall, you need to review and modify the firewall to match the new behavior. This change was introduced into Junos OS Evolved versions: • 21.4R2-S1-EVO, 21.4R2-S2-EVO, 21.4R3-S1-EVO and later • Exclude 21.4R3-EVO • 22.2R2-EVO • 22.3R1-EVO Match the first 8-bit Next Header field in the packet. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstops` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `mobility` (135), `no-next-header` (59), `ospf` (89), `pim` (103), `routing` (43), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112). **NOTE**: • `next-header icmp6` and `next-header icmpv6` match conditions perform the same function. `next-header icmp6` is the preferred option. `next-header icmpv6` is hidden in the Junos OS CLI. • On the QFX5000 series devices running Junos OS Evolved, the `next-header` match is not supported under ERACLv6, and instead, you must configure the `payload-protocol` match.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`next-header-except` *header-type*	Do not match the 8-bit Next Header field that identifies the type of header between the IPv6 header and payload. For details, see the `next-header` match type.
`packet-length` *bytes*	Match the 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` *bytes*	Do not match the length of the received packet, in bytes. For details, see the `packet-length` match type.
`payload-protocol` *protocol-type*	Match the payload protocol type. In place of the *protocol-type* numeric value, you can specify one of the following text synonyms (the field values are also listed): specify one or a set of of the following: `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58, `igmp` (2), `ipip` (4), `ipv6` (41), `no-next-header`, `ospf` (89), `pim` (103), `routing`, `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112) (dstopts (60), fragment (44), hop-by-hop 0), and routing are not available in Junos OS Release 16.1 and later). You can also use the `payload-protocol` condition to match an extension header type that the Juniper Networks firmware cannot interpret. You can specify a range of extension header values within square brackets. When the firmware finds the first extension header type that it cannot interpret in a packet, the `payload-protocol` value is set to that extension header type. The firewall filter only examines the first extension header type that the firmware cannot interpret in the packet. **NOTE**: This match condition is only supported on MPCs on MX Series Routers. Initialize new firewall filters that include this condition by walking the corresponding SNMP MIB.
`payload-protocol-except` *protocol-type*	Do not match the payload protocol type. For details, see the `payload-protocol` match type. **NOTE**: This match condition is only supported on MPCs on MX Series Routers

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`port` *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the `destination-port` match condition or the `source-port` match condition in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. **NOTE**: For Junos OS Evolved, you must configure the `next-header` match statement in the same term. In place of the numeric value, you can specify one of the text synonyms listed under `destination-port`.
`port-except` *number*	Do not match the UDP or TCP source or destination port field. For details, see the `port` match condition.
`prefix-list` *prefix-list-name* `[ except ]`	Match the prefixes of the source or destination address fields to the prefixes in the specified list unless the `except` option is included. If the option is included, do not match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the `[edit policy-options prefix-list` *prefix-list-name*`]` hierarchy level.
`service-filter-hit`	Match a packet received from a filter where a `service-filter-hit` action was applied.
`source-address` *address* `[ except ]`	Match the IPv6 address of the source node sending the packet unless the `except` option is included. If the option is included, do not match the IPv6 address of the source node sending the packet. You cannot specify both the `address` and `source-address` match conditions in the same term.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`source-class` *`class-names`*	Match one or more specified source class names (sets of source prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 992.
`source-class-except` *`class-names`*	Do not match one or more specified source class names. For details, see the `source-class` match condition.
`source-port` *`number`*	Match the UDP or TCP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. **NOTE**: For Junos OS Evolved, you must configure the `next-header` or `next-header tcp` match statement in the same term. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *`number`* match condition.
`source-port-except` *`number`*	Do not match the UDP or TCP source port field. For details, see the `source-port` match condition.
`source-prefix-list` *`name`* `[ except ]`	Match the IPv6 address prefix of the packet source field unless the `except` option is included. If the option is included, do not match the IPv6 address prefix of the packet source field. Specify a prefix list name defined at the `[edit policy-options prefix-list` *`prefix-list-name`*`]` hierarchy level.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`tcp-established`	Match TCP packets other than the first packet of a connection. This is a text synonym for `tcp-flags "(ack | rst)"` (`0x14`). **NOTE**: This condition does not implicitly check that the protocol is TCP. To check this, specify the `protocol tcp` match condition. If you configure this match condition, we recommend that you also configure the `next-header tcp` match condition in the same term.
`tcp-flags` *`flags`*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the `tcp-established` and `tcp-initial` match conditions. If you configure this match condition, we recommend that you also configure the `next-header tcp` match condition in the same term to specify that the TCP protocol is being used on the port.

Table 52: Firewall Filter Match Conditions for IPv6 Traffic *(Continued)*

Match Condition	Description
`tcp-initial`	Match the initial packet of a TCP connection. This is a text synonym for `tcp-flags "(! ack & syn)"`. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the `next-header tcp` match condition in the same term.

`traffic-class` *number*	Match the 8-bit field that specifies the class-of-service (CoS) priority of the packet.
	
	This field was previously used as the type-of-service (ToS) field in IPv4.
	
	You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46).
	
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	
	  • `af11` (10), `af12` (12), `af13` (14)
	
	  • `af21` (18), `af22` (20), `af23` (22)
	
	  • `af31` (26), `af32` (28), `af33` (30)
	
	  • `af41` (34), `af42` (36), `af43` (38)

`traffic-class-except` *number*	Do not match the 8-bit field that specifies the CoS priority of the packet. For details, see the `traffic-class` match description.

NOTE: If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see IPv6 Overview and Supported IPv6 Standards.

Release History Table

Release	Description
13.3R6	Support for the `next-header` firewall match condition is available in Junos OS Release 13.3R6 and later.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Terminating Actions | **884**
- Firewall Filter Nonterminating Actions | **871**
- Firewall Filter Match Conditions for IPv4 Traffic | **940**
- *enhanced-mode*
- Firewall Filter Flexible Match Conditions | **862**

Firewall Filter Match Conditions Based on Numbers or Text Aliases

IN THIS SECTION

- Matching on a Single Numeric Value | **975**
- Matching on a Range of Numeric Values | **975**
- Matching on a Text Alias for a Numeric Value | **975**
- Matching on a List of Numeric Values or Text Aliases | **975**

Matching on a Single Numeric Value

You can specify a *firewall filter* match condition based on whether a particular packet field value is a specified numeric value. In the following example, a match occurs if the packet source port number is **25**:

```
[edit firewall family inet filter filter1 term term1 from]
user@host# set source-port 25
```

Matching on a Range of Numeric Values

You can specify a firewall filter match condition based on whether a particular packet field value falls within a specified range of numeric values. In the following example, a match occurs for source ports values from **1024** through **65,535**, inclusive:

```
[edit firewall family inet filter filter2 term term1 from]
user@host# set source-port 1024-65536
```

Matching on a Text Alias for a Numeric Value

You can specify a firewall filter match condition based on whether a particular packet field value is a numeric value that you specify by using a text string as an *alias* for the numeric value. In the following example, a match occurs if the packet source port number is **25**. For the **source-port** and **destination-port** match conditions, the text alias**smtp** corresponds to the numeric value **25**.

```
[edit firewall family inet filter filter3 term term1 from]
user@host# set source-port smtp
```

Matching on a List of Numeric Values or Text Aliases

You can specify a firewall filter match condition based on whether a particular packet field value matches any one of multiple numeric values or text aliases that you specify within square brackets and delimited by spaces. In the following example, a match occurs if the packet source port number is any of

the following values: **20** (which corresponds to the text aliases **ftp-data**), **25**, or any value from **1024** through **65535**.

```
[edit firewall family inet filter filter3 term term1 from]
user@host# set source-port [ smtp ftp-data 25 1024-65535 ]
```

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Match Conditions Based on Bit-Field Values | **976**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**

Firewall Filter Match Conditions Based on Bit-Field Values

IN THIS SECTION

- Match Conditions for Bit-Field Values | **976**
- Match Conditions for Common Bit-Field Values or Combinations | **977**
- Logical Operators for Bit-Field Values | **978**
- Matching on a Single Bit-Field Value or Text Alias | **979**
- Matching on Multiple Bit-Field Values or Text Aliases | **980**
- Matching on a Negated Bit-Field Value | **980**
- Matching on the Logical OR of Two Bit-Field Values | **981**
- Matching on the Logical AND of Two Bit-Field Values | **981**
- Grouping Bit-Field Match Conditions | **981**

Match Conditions for Bit-Field Values

Table 53 on page 977 lists the *firewall filter* match conditions that are based on whether certain bit fields in a packet are set or not set. The second and third columns list the types of traffic for which the match condition is supported.

Table 53: Binary and Bit-Field Match Conditions for Firewall Filters

Bit-Field Match Condition	Match Values	Protocol Families for Standard Stateless Firewall Filters	Protocol Families for Service Filters
fragment-flags *flags*	Hexadecimal values or text aliases for the three-bit IP fragmentation flags field in the IP header.	**family inet**	**family inet**
fragment-offset *value*	Hexadecimal values or text aliases for the 13-bit fragment offset field in the IP header.	**family inet**	**family inet**
tcp-flags *value*†	Hexadecimal values or text aliases for the low-order 6 bits of the 8-bit TCP flags field in the TCP header.	**family inet** **family inet6** **family vpls** **family bridge**	**family inet** **family inet6**

† The Junos OS does not automatically check the first fragment bit when matching TCP flags for IPv4 traffic. To check the first fragment bit for IPv4 traffic only, use the **first-fragment** match condition.

Match Conditions for Common Bit-Field Values or Combinations

Table 54 on page 978 describes firewall filter match conditions that are based on whether certain commonly used values or *combinations* of bit fields in a packet are set or not set.

You can use text synonyms to specify some common bit-field matches. In the previous example, you can specify **tcp-initial** as the same match condition.

NOTE: Some of the numeric range and bit-field match conditions allow you to specify a text synonym. For a complete list of synonyms:

- If you are using the J-Web interface, select the synonym from the appropriate list.
- If you are using the CLI, type a question mark (**?**) after the **from** statement.

Table 54: Bit-Field Match Conditions for Common Combinations

Match Condition	Description	Protocol Families for Standard Stateless Firewall Filters	Protocol Families for Service Filters
first-fragment	Text alias for the bit-field match condition **fragment-offset 0**, which indicates the first fragment of a fragmented packet.	**family inet**	**family inet**
is-fragment	Text alias for the bit-field match condition **fragment-offset 0 except**, which indicates a trailing fragment of a fragmented packet.	**family inet**	**family inet**
tcp-established	Alias for the bit-field match condition **tcp-flags "(ack | rst)"**, which indicates an established TCP session, but not the first packet of a TCP connection.	**family inet** **family inet6**	—
tcp-initial	Alias for the bit-field match condition **tcp-flags "(!ack & syn)"**, which indicates the first packet of a TCP connection, but not an established TCP session.	**family inet** **family inet6**	—

Logical Operators for Bit-Field Values

Table 55 on page 979 lists the logical operators you can apply to *single* bit-field values when specifying stateless firewall filter match conditions. The operators are listed in order, from highest precedence to lowest precedence. Operations are left-associative, meaning that the operations are processed from left to right.

Table 55: Bit-Field Logical Operators

Precedence Order	Bit-Field Logical Operator	Description
1	**(***complex-match-condition***)**	Grouping—The complex match condition is evaluated before any operators outside the parentheses are applied.
2	**!** *match-condition*	Negation—A match occurs if the match condition is false.
3	*match-condition-1* **&** *match-condition-2* or *match-condition-1* **+** *match-condition-2*	Logical AND—A match occurs if both match conditions are true.
4	*match-condition-1* **|** *match-condition-2* or *match-condition-1* **,** *match-condition-2*	Logical OR—A match occurs if either match condition is true.

Matching on a Single Bit-Field Value or Text Alias

For the **fragment-flags** and **tcp-flags** bit-match conditions, you can specify firewall filter match conditions based on whether a particular bit in the packet field is set or not set.

- Numeric value to specify a single bit—You can specify a single bit-field match condition by using a numeric value that has one bit set. Depending on the match condition, you can specify a decimal value, a binary value, or a hexadecimal value. To specify a binary value, specify the number with the prefix **b**. To specify a hexadecimal value, specify the number with the prefix **0x**.

  In the following example, a match occurs if the **RST** bit in the TCP flags field is set:

  ```
  [edit firewall family inet filter filter_tcp_rst_number term term1 from]
  user@host# set tcp-flags 0x04
  ```

- Text alias to specify a single bit—You generally specify a single bit-field match condition by using a text alias enclosed in double-quotation marks (“ ”).

In the following example, a match occurs if the **RST** bit in the TCP flags field is set:

```
[edit firewall family inet filter filter_tcp_rst_alias term term1 from]
user@host# set tcp-flags “rst”
```

Matching on Multiple Bit-Field Values or Text Aliases

You can specify a firewall filter match condition based on whether a particular set of bits in a packet field are set.

- Numeric values to specify multiple set bits—When you specify a numeric value whose binary representation has more than one set bit, the value is treated as a logical AND of the set bits.

  In the following example, the two match conditions are the same. A match occurs if either bit **0x01** or **0x02** is not set:

  ```
  [edit firewall family inet filter reset_or_not_initial_packet term term5 from]
  user@host# set tcp-flags “!0x3”
  user@host# set tcp-flags “!(0x01 & 0x02)”
  ```

- Text aliases that specify common bit-field matches—You can use text aliases to specify some common bit-field matches. You specify these matches as a single keyword.

  In the following example, the **tcp-established** condition, which is an alias for **“(ack | rst)”**, specifies that a match occurs on TCP packets other than the first packet of a connection:

  ```
  [edit firewall family inet filter reset_or_not_initial_packet term term6 from]
  user@host# set tcp-established
  ```

Matching on a Negated Bit-Field Value

To negate a match, precede the value with an exclamation point.

In the following example, a match occurs if the **RST** bit in the TCP flags field is set:

```
[edit firewall family inet filter filter_tcp_rst term term1 from]
user@host# set tcp-flags “!rst”
```

Matching on the Logical OR of Two Bit-Field Values

You can use the (**|** or **,**) to specify that a match occurs if a bit field matches either of two bit-field values specified.

In the following example, a match occurs if the packet is *not* the initial packet in a TCP session:

```
[edit firewall family inet filter not_initial_packet term term3 from]
user@host# set tcp-flags "!syn | ack"
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is not the initial packet in a TCP session, either the SYN flag is not set or the ACK flag is set.

Matching on the Logical AND of Two Bit-Field Values

You can use the (**&** or **+**) to specify that a match occurs if a bit field matches both of two bit-field values specified.

In the following example, a match occurs if the packet is the initial packet in a TCP session:

```
[edit firewall family inet filter initial_packet term term2 from]
user@host# set tcp-flags “syn & !ack”
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is an initial packet in a TCP session, the SYN flag is set and the ACK flag is not set.

Grouping Bit-Field Match Conditions

You can use the to specify that the complex match condition inside the parentheses is evaluated before any operators outside the parentheses are applied.

In the following example, a match occurs if the packet is a TCP reset or if the packet is not the initial packet in the TCP session:

```
[edit firewall family inet filter reset_or_not_initial_packet term term4 from]
user@host# set tcp-flags “!(syn & !ack) | rst”
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is *not* the initial packet in a TCP session, the SYN flag is not set and the ACK field is set.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**

Firewall Filter Match Conditions Based on Address Fields

IN THIS SECTION

- Implied Match on the '0/0 except' Address for Firewall Filter Match Conditions Based on Address Fields | **982**
- Matching an Address Field to a Subnet Mask or Prefix | **983**
- Matching an Address Field to an Excluded Value | **984**
- Matching Either IP Address Field to a Single Value | **989**
- Matching an Address Field to Noncontiguous Prefixes | **989**
- Matching an Address Field to a Prefix List | **991**

You can configure *firewall filter* match conditions that evaluate packet address fields—IPv4 source and destination addresses, IPv6 source and destination addresses, or media access control (MAC) source and destination addresses—against specified addresses or prefix values.

Implied Match on the '0/0 except' Address for Firewall Filter Match Conditions Based on Address Fields

Every firewall filter match condition based on a set of addresses or address prefixes is associated with an implicit match on the address **0.0.0.0/0 except** (for IPv4 or VPLS traffic) or **0:0:0:0:0:0:0:0/0 except** (for IPv6 traffic). As a result, any packet whose specified address field does not match any of the specified addresses or address prefixes fails to match the entire term.

Matching an Address Field to a Subnet Mask or Prefix

You can specify a single match condition to match a source address or destination address that falls within a specified address prefix.

IPv4 Subnet Mask Notation

For an IPv4 address, you can specify a subnet mask value rather than a prefix length. For example:

```
[edit firewall family inet filter filter_on_dst_addr term term3 from]
user@host# set address 10.0.0.10/255.0.0.255
```

Prefix Notation

To specify the address prefix, use the notation ***prefix*/*prefix-length***. In the following example, a match occurs if a destination address matches the prefix **10.0.0.0/8**:

```
[edit firewall family inet filter filter_on_dst_addr term term1 from]
user@host# set destination-address 10.0.0.0/8
```

Default Prefix Length for IPv4 Addresses

If you do not specify ***/prefix-length*** for an IPv4 address, the prefix length defaults to **/32**. The following example illustrates the default prefix value:

```
[edit firewall family inet filter filter_on_dst_addr term term2 from]
user@host# set destination-address 10
user@host# show
destination-address {
    10.0.0.0/32;
}
```

Default Prefix Length for IPv6 Addresses

If you do not specify */prefix-length* for an IPv6 address, the prefix length defaults to **/128**. The following example illustrates the default prefix value:

```
[edit firewall family inet6 filter filter_on_dst_addr term term1 from]
user@host# set destination-address ::10
user@host# show
destination-address {
    ::10/128;
}
```

Default Prefix Length for MAC Addresses

If you do not specify */prefix-length* for a media access control (MAC) address of a VPLS, Layer 2 CCC, or Layer 2 bridging packet, the prefix length defaults to **/48**. The following example illustrates the default prefix value:

```
[edit firewall family vpls filter filter_on_dst_mac_addr term term1 from]
user@host# set destination-mac-address 01:00:0c:cc:cc:cd
user@host# show
destination-address {
    01:00:0c:cc:cc:cd/48;
}
```

Matching an Address Field to an Excluded Value

For the address-field match conditions, you can include the **except** keyword to specify that a match occurs for an address field that does not match the specified address or prefix.

Excluding IP Addresses in IPv4 or IPv6 Traffic

For the following IPv4 and IPv6 match conditions, you can include the **except** keyword to specify that a match occurs for an IP address field that does not match the specified IP address or prefix:

- **address** *address* **except**—A match occurs if either the source IP address or the destination IP address does not match the specified address or prefix.
- **source-address** *address* **except**—A match occurs if the source IP address does not match the specified address or prefix.

- **destination-address *address* except**—A match occurs if the destination IP address does not match the specified address or prefix.

In the following example, a match occurs for any IPv4 destination addresses that fall under the **172.0.0.0/8** prefix, except for addresses that fall under **172.16.0.0/16**. All other addresses implicitly do not match this condition.

```
[edit firewall family inet filter filter_on_dst_addr term term1 from]
user@host# set destination-address 172.16.0.0/16 except
user@host# set destination-address 172.0.0.0/8
user@host# show
destination-address {
    172.16.0.0/16 except;
    172.0.0.0/8;
}
```

In the following example, a match occurs for any IPv4 destination address that does not fall within the prefix **10.1.1.0/24**:

```
[edit firewall family inet filter filter_on_dst_addr term term24 from]
user@host# set destination-address 0.0.0.0/0
user@host# set destination-address 10.1.1.0/24 except
user@host# show
destination-address {
    0.0.0.0/0;
    10.1.1.0/24 except;
}
```

Excluding IP Addresses in VPLS or Layer 2 Bridging Traffic

For the following VPLS and Layer 2 bridging match conditions on MX Series routers only, you can include the **except** keyword to specify that a match occurs for an IP address field that does not match the specified IP address or prefix:

- **ip-address *address* except**—A match occurs if either the source IP address or the destination IP address does not match the specified address or prefix.
- **source-ip-address *address* except**—A match occurs if the source IP address does not match the specified address or prefix.
- **destination-ip-address *address* except**—A match occurs if the destination IP address does not match the specified address or prefix.

In the following example for filtering VPLS traffic on an MX Series router, a match occurs if the source IP address falls within the exception range of **55.0.1.0/255.0.255.0** and the destination IP address matches **5172.16.5.0/8**:

```
[edit]
firewall {
    family vpls {
        filter fvpls {
            term 1 {
                from {
                    ip-address {
                        55.0.0.0/8;
                        55.0.1.0/255.0.255.0 except;
                    }
                }
                then {
                    count from-55/8;
                    discard;
                }
            }
        }
    }
}
```

Excluding MAC Addresses in VPLS or Layer 2 Bridging Traffic

For the following VPLS or Layer 2 bridging traffic match conditions, you can include the **except** keyword to specify that a match occurs for a MAC address field that does not match the specified MAC address or prefix:

- **source-mac-address *address* except**—A match occurs if the source MAC address does not match the specified address or prefix.
- **destination-mac-address *address* except**—A match occurs if either the destination MAC address does not match the specified address or prefix.

Excluding All Addresses Requires an Explicit Match on the ’0/0’ Address

If you specify a firewall filter match condition that consists of one or more address-*exception* match conditions (address match conditions that use the **except** keyword) but no *matchable* address match conditions, packets that do not match any of the configured prefixes fails the overall match operation. To configure a firewall filter term of address-exception match conditions to match any address that is not in the prefix list, include an explicit match of **0/0** so that the term contain a matchable address.

For the following example firewall filter for IPv4 traffic, the **from-trusted-addresses** term fails to discard matching traffic, and the **INTRUDERS-COUNT** counter is missing from the output of the **show firewall** *operational mode command*:

```
[edit]
user@host# show policy-options
prefix-list TRUSTED-ADDRESSES {
    10.2.1.0/24;
    192.168.122.0/24;
}

[edit firewall family inet filter protect-RE]
user@host# show
term from-trusted-addresses {
    from {
        source-prefix-list {
            TRUSTED-ADDRESSES except;
        }
        protocol icmp;
    }
    then {
        count INTRUDERS-COUNT;
        discard;
    }
}
    term other-icmp {
        from {
            protocol icmp;
        }
        then {
            count VALID-COUNT;
            accept;
        }
    }
```

```
    term all {
        then accept;
    }
```

```
[edit]
user@host# run show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
VALID-COUNT                                          2770                   70
Filter: __default_bpdu_filter__
```

To cause a filter term of address-exception match conditions to match any address that is not in the prefix list, include an explicit match of **0/0** in the set of match conditions:

```
[edit firewall family inet filter protect-RE]
user@host# show term from-trusted-addresses
from {
    source-prefix-list {
        0.0.0.0/0;
        TRUSTED-ADDRESSES except;
    }
    protocol icmp;
}
```

With the addition of the **0.0.0.0/0** source prefix address to the match condition, the **from-trusted-addresses** term discards matching traffic, and the INTRUDERS-COUNT counter displays in the output of the **show firewall** operational mode command:

```
[edit]
user@host# run show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
VALID-COUNT                                          2770                   70
INTRUDERS-COUNT                                       420                    5
Filter: __default_bpdu_filter__
```

Matching Either IP Address Field to a Single Value

For IPv4 and IPv6 traffic and for VPLS and Layer 2 bridging traffic on MX Series routers only, you can use a single match condition to match a single address or prefix value to either the source or destination IP address field.

Matching Either IP Address Field in IPv4 or IPv6 Traffic

For IPv4 or IPv6 traffic, you can use a single match condition to specify the same address or prefix value as the match for either the source or destination IP address field. Instead of creating separate filter terms that specify the same address for the **source-address** and **destination-address** match conditions, you use only the **address** match condition. A match occurs if *either* the source IP address *or* the destination IP address matches the specified address or prefix.

If you use the **except** keyword with the **address** match condition, a match occurs if *both* the source IP address and the destination IP address match the specified value *before* the exception applies.

In a firewall filter term that specifies either the **source-address** or the **destination-address** match condition, you cannot also specify the **address** match condition.

Matching Either IP Address Field in VPLS or Layer 2 Bridging Traffic

For VPLS or Layer 2 bridging traffic on MX Series routers only, you can use a single match condition to specify the same address or prefix value as the match for either the source or destination IP address field. Instead of creating separate filter terms that specify the same address for the **source-ip-address** and **destination-ip-address** match conditions, you use only the **ip-address** match condition. A match occurs if *either* the source IP address *or* the destination IP address matches the specified address or prefix.

If you use the **except** keyword with the **ip-address** match condition, a match occurs if *both* the source IP address and the destination IP address match the specified value *before* the exception applies.

In a firewall filter term that specifies either the **source-ip-address** or the **destination-ip-address** match condition, you cannot also specify the **ip-address** match condition.

Matching an Address Field to Noncontiguous Prefixes

For IPv4 traffic only, specify a single match condition to match the IP source or destination address field to any prefix specified. The prefixes do not need to be contiguous. That is, the prefixes under the **source-address** or **destination-address** match condition do not need to be adjacent or neighboring to one another.

In the following example, a match occurs if a destination address matches either the **10.0.0.0/8** prefix or the **192.168.0.0/32** prefix:

```
[edit firewall family inet filter filter_on_dst_addr term term5 from]
user@host# set destination-address 10.0.0.0/8
user@host# set destination-address 192.168.0.0/32
user@host# show
destination-address {
    destination-address 10.0.0.0/8;
    destination-address 192.168.0.0/32;
}
```

The order in which you specify the prefixes within the match condition is not significant. Packets are evaluated against all the prefixes in the match condition to determine whether a match occurs. If prefixes overlap, longest-match rules are used to determine whether a match occurs. A match condition of noncontiguous prefixes includes an implicit **0/0 except** statement, which means that any prefix that does not match any prefix included in the match condition 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 inet filter filter_on_src_addr term term1 from]
source-address {
    172.16.0.0/10;
    172.16.2.0/24 except;
    192.168.1.0;
    192.168.1.192/26 except;
    192.168.1.254;
    172.16.3.0/24; # ignored
    10.2.2.2 except; # ignored
}
```

Within the **source-address** match condition, two addresses are ignored. The **172.16.3.0/16** value is ignored because it falls under the address **172.16.0.0/10**, which is the same type. The **10.2.2.2 except** value is ignored because it is subsumed by the implicit **0.0.0.0/0 except** match value.

Suppose the following source IP address are evaluated by this firewall filter:

- Source IP address **172.16.1.2**—This address matches the **172.16.0.0/10** prefix, and thus the action in the **then** statement is taken.

- Source IP address **172.16.2.2**—This address matches the **172.16.2.0/24** prefix. Because this prefix is negated (that is, includes the **except** keyword), 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.
- Source IP address **10.1.2.3**—This address 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 of prefixes configured under the **source-address** match condition, and is considered to be a mismatch.

  The **172.16.3.0/24** 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 of prefixes configured under the **source-address** match condition.

> **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 might be processed by the latter term before they are evaluated by any intervening terms. As a result, packets that should be rejected by the intervening terms might be accepted instead, or packets that should be accepted might be rejected instead.
>
> 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.

Matching an Address Field to a Prefix List

You can define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or in a stateless firewall filter match condition that evaluates packet address fields.

To define a list of IPv4 or IPv6 address prefixes, include the **prefix-list** *prefix-list* statement.

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

You can include the statement at the following hierarchy levels:

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

After you have defined a prefix list, you can use it when specifying a firewall filter match condition based on an IPv4 or IPv6 address prefix.

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

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | 814
- Firewall Filter Match Conditions Based on Numbers or Text Aliases | 974
- Firewall Filter Match Conditions Based on Bit-Field Values | 976
- Firewall Filter Match Conditions Based on Address Classes | 992

Firewall Filter Match Conditions Based on Address Classes

IN THIS SECTION

- Source-Class Usage | 993
- Destination-Class Usage | 993
- Guidelines for Applying SCU or DCU Firewall Filters to Output Interfaces | 993

For IPv4 and IPv6 traffic only, you can use class-based *firewall filter* conditions to match packet fields based on source class or destination class.

Source-Class Usage

A is a set of source prefixes grouped together and given a class name. To configure a firewall filter term that matches an IP source address field to one or more source classes, use the `source-class` *`class-name`* match condition under the `[edit firewall family (inet | inet6) filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

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.

Destination-Class Usage

A is a set of destination prefixes grouped together and given a class name. To configure a firewall filter term that matches an IP destination address field to one or more destination classes, use the `destination-class` *`class-name`* match condition at the `[edit firewall family (inet | inet6) filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

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.

Note, however, that 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.

Guidelines for Applying SCU or DCU Firewall Filters to Output Interfaces

When applying a SCU or DCU firewall filter to an interface, keep the following guidelines in mind:

- Output interfaces—Class-based firewall filter match conditions work only for firewall filters that you apply to output interfaces. This is because the SCU and DCU are determined after route lookup occurs.
- Input interfaces—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.
- Output interfaces for tunnel traffic—SCU and DCU are not supported on the interfaces you configure as the output interface for tunnel traffic for transit packets exiting the router (or switch) through the tunnel.

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters | **814**

Firewall Filter Match Conditions for IPv4 Traffic | **940**

Firewall Filter Match Conditions for IPv6 Traffic | 957
Junos OS Routing Policies, Firewall Filters, and Traffic Policers User Guide
Firewall Filter Match Conditions Based on Numbers or Text Aliases | 974
Firewall Filter Match Conditions Based on Bit-Field Values | 976
Firewall Filter Match Conditions Based on Address Fields | 982

Understanding IP-Based Filtering and Selective Port Mirroring of MPLS Traffic

IN THIS SECTION

- IP-Based Filtering of MPLS Traffic | 994
- Selective Port Mirroring of MPLS Traffic | 995
- Sample Configurations | 996

In an MPLS packet, the IP header comes immediately after the MPLS header. The IP-based filtering feature provides a deep inspection mechanism, where a maximum of upto eight MPLS labels of the inner payload can be inspected to enable filtering of MPLS traffic based on IP parameters. The filtered MPLS traffic can also be port mirrored to a monitoring device to offer network-based services in the core MPLS network.

IP-Based Filtering of MPLS Traffic

Prior to Junos OS Release 18.4R1, filtering based on IP parameters was not supported for MPLS family filter. With the introduction of the IP-based filtering feature, you can apply inbound and outbound filters for MPLS-tagged IPv4 and IPv6 packets based on IP parameters, such as source and destination addresses, Layer 4 protocol type, and source and destination ports.

The IP-based filtering feature enables you to filter MPLS packets at the ingress of an interface, where the filtering is done using match conditions on the inner payload of the MPLS packet. The selective MPLS traffic can then be port mirrored to a remote monitoring device using logical tunnels.

To support IP-based filtering, additional match conditions are added that allow MPLS packets to be deep inspected to parse the inner payload with Layer 3 and Layer 4 headers before the appropriate filters are applied.

NOTE: The IP-based filtering feature is supported only for MPLS-tagged IPv4 and IPv6 packets. In other words, the MPLS filters match IP parameters only when the IP payload comes immediately after the MPLS labels.

In other scenarios, where the MPLS payload includes pseudowires, protocols other than inet and inet6, or other encapsulations like Layer 2 VPN or VPLS, the IP-based filtering feature is not supported.

The following match conditions are added for the IP-based filtering of MPLS traffic:

- IPv4 source address
- IPv4 destination address
- IPv6 source address
- IPv6 destination address
- Protocol
- Source port
- Destination port
- Source IPv4 prefix list
- Destination IPv4 prefix list
- Source IPv6 prefix list
- Destination IPv6 prefix list

NOTE: The following match combinations are supported for the IP-based filtering of MPLS traffic:

- Source and destination address match conditions with IPv4 and IPv6 prefix lists.
- Source and destination port address and protocol types match conditions with IPv4 and IPv6 prefix lists.

Selective Port Mirroring of MPLS Traffic

Port mirroring is the capability of mirroring a packet to a configured destination, in addition to the normal processing and forwarding of the packets. Port mirroring is applied as an action for a firewall

filter, which is applied at the ingress or egress of any interface. Similarly, the selective port mirroring feature provides the capability to mirror MPLS traffic, which is filtered based on IP parameters, to a mirrored destination using logical tunnels.

To enable selective port mirroring, additional actions are configured at the `[edit firewall family mpls filter filter-name term term-name then]` hierarchy level, in addition to the existing `counter`, `accept`, and `discard` actions:

- `port-mirror`
- `port-mirror-instance`

Port Mirroring

The `port-mirror` action enables port mirroring globally on the device, which applies to all Packet Forwarding Engines (PFEs) and associated interfaces.

For MPLS family filter, the `port-mirror` action is enabled for global port mirroring.

Port Mirroring Instance

The `port-mirror-instance` action enables you to customize each instance with different properties for input sampling and port mirroring output destinations, instead of having to use a single system-wide configuration for port mirroring.

You can configure only two port mirroring instances per Flexible PIC Concentrator (FPC) by including the `instance port-mirror-instance-name` statement at the `[edit forwarding-options port-mirror]` hierarchy level. You can then associate individual port mirroring instances with an FPC, PIC, or (Forwarding Engine Board (FEB) depending on the device hardware.

For MPLS family filter, the `port-mirror-instance` action is enabled only for the port-mirroring instance.

NOTE: For both `port-mirror` and `port-mirror-instance` actions, the output interface must be enabled with Layer 2 family and not family MPLS (Layer 3) for the selective port mirroring feature to work.

Sample Configurations

IP-Based Filtering Configuration

```
[edit firewall family mpls filter mpls-filter]
term ipv4-term {
    from {
```

```
        ip-version {
            ipv4 {
                source-address {
                    10.10.10.10/24;
                }
                destination-address {
                    20.20.20.20/24;
                }
                protocol tcp {
                    source-port 100;
                    destination-port 200;
                }
                soure-prefix-list ipv4-source-users;
                destination-prefix-list ipv4-destination-users;
            }
        }
        exp 1;
        }
        then port-mirror;
        then accept;
    then count;
}
term ipv6-term {
    from {
        ip-version {
            ipv6 {
                source-address {
                    2000::1/128;
                }
                destination-address {
                    3000::1/128;
                }
                protocol tcp {
                    source-port 100;
                    destination-port 200;
                }
                source-prefix-list ipv6-source-users;
                destination-prefix-list ipv6-destination-users;
            }
        }
        exp 1;
    }
    then port-mirror-instance port-mirror-instance1;
```

```
    then accept;
    then count;
}
```

```
[edit policy-options]
prefix-list ipv4-source-users {
    172.16.1.16/28;
    172.16.2.16/28;
}
prefix-list ipv6-source-users {
    2001::1/128;
    3001::1/128;
}
```

```
[edit interfaces]
xe-0/0/1 {
    unit 0 {
        family inet {
            address 100.100.100.1/30;
        }
        family mpls {
            filter {
                input mpls-filter;
            }
        }
    }
}
```

Selective Port Mirroring Configuration

```
[edit forwarding-options]
port-mirroring {
    input {
        rate 2;
        run-length 4;
        maximum-packet-length 500;
    }
    family any {
```

```
        output {
            interface xe-2/0/2.0;
        }
    }
}
```

```
[edit forwarding-options]
port-mirroring {
    instance {
        port-mirror-instance1 {
            input {
                rate 3;
                run-length 5;
                maximum-packet-length 500;
            }
            family any {
                output {
                    interface xe-2/0/2.0;
                }
            }
        }
    }
}
```

NOTE: The output interface `xe-2/0/2.0` is configured for Layer 2 family and not family MPLS. For both `port-mirror` and `port-mirror-instance` actions, the output interface must be enabled with Layer 2 family and not family MPLS (Layer 3) for the selective port mirroring feature to work.

Mirrored Destination Configuration

```
[edit interfaces]
xe-2/0/2 {
    vlan-tagging;
    encapsulation extended-vlan-bridge;
    unit 0 {
        vlan-id 600;
```

```
    }
}
```

```
[edit bridge-domains]
bd {
    domain-type bridge;
    interface xe-2/0/2.0;
}
```

Firewall Filter Match Conditions for MPLS Traffic

You can configure a firewall filter with match conditions for MPLS traffic (`family mpls`).

- The `input-list` *`filter-names`* and `output-list` *`filter-names`* statements for firewall filters for the `mpls` protocol family are supported on all interfaces except for management interfaces and internal Ethernet interfaces (`fxp` or `em0`), loopback interfaces (`lo0`), and USB modem interfaces (`umd`)
- (QFX5100, QFX5110, QFX5200, QFX5210) If you are applying an MPLS filter on a loopback interface, you can only filter on the `label`, `exp`, `ttl=1`, and Layer 4 `tcp` and `udp` port number fields. For TTL, you must explicitly specify `ttl=1` under `family mpls` to match on TTL=1 packets. The only actions you can configure are `accept`, `discard`, and `count`. You can apply the filter only in the ingress direction.
- For MX Series Routers with MPC and MIC, you can apply inbound and outbound filters for MPLS family based on MPLS-tagged IPv4 and IPv6 parameters using inner payload match conditions, and enable selective port mirroring of MPLS traffic unto a monitoring device (starting in Junos OS Release 18.4R1). For IP-based filtering, additional match conditions are available under the MPLS filter term `from` parameter, and to support port mirroring, additional actions (such as port-mirror and port-mirror-instance), are available under the filter term `then`parameter.

Table 56 on page 1000 describes the *`match-conditions`* you can configure at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 56: Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
`apply-groups`	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.
`destination-port` *number*	Match on the UDP or TCP destination port field. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`exp` *number*	Experimental (EXP) bit number or range of bit numbers in the MPLS header of a packet. For *number*, you can specify one or more values from 0 through 7 in binary, decimal or hexadecimal format, as given below: • A single EXP bit—for example, **exp 3** • Several EXP bits—for example, **exp 0,4** • A range of EXP bits—for example, **exp [0-5]**. These values are not supported on filters applied to the loopback interface. **NOTE**: This match condition is deprecated on PTX10001-36MR, PTX10003, PTX10004, PTX10008, and PTX10016 devices and is replaced by `exp0` *number*.
`exp-except` *number*	Do not match on the EXP bit number or range of bit numbers in the MPLS header. For *number*, you can specify one or more values from `0` through `7`. **NOTE**: This match condition is deprecated on PTX10001-36MR, PTX10003, PTX10004, PTX10008, and PTX10016 devices and is replaced by `exp0-except`.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`exp0` *number*	Experimental (EXP) bit number or range of bit numbers in the TOS MPLS header of a packet. For *number*, you can specify one or more values from 0 through 7 in binary, decimal or hexadecimal format, as given below: • A single EXP bit—for example, **exp0 3** • Several EXP bits—for example, **exp0 0,4** • A range of EXP bits—for example, **exp0 [0-5]**. These values are not supported on filters applied to the loopback interface.
`exp0-except` *number*	Do not match EXP bit number or range of bit numbers in the TOS MPLS header of a packet. For *number*, you can specify one or more values from 0 through 7 in binary, decimal or hexadecimal format, as given below: • A single EXP bit—for example, **exp0-except 3** • Several EXP bits—for example, **exp0-except 0,4** • A range of EXP bits—for example, **exp0-except [0-5]**. These values are not supported on filters applied to the loopback interface.
`exp1` *number*	Experimental (EXP) bit number or range of bit numbers in the MPLS header that is next to the TOS (top of stack) MPLS header. For *number*, you can specify one or more values from 0 through 7 in binary, decimal or hexadecimal format, as given below: • A single EXP bit—for example, **exp1 3** • Several EXP bits—for example, **exp1 0,4** • A range of EXP bits—for example, **exp1 [0-5]**. These values are not supported on filters applied to the loopback interface.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`exp1-except` *number*	Do not match on the EXP bit number or range of bit numbers in the MPLS header next to the TOS MPLS header. For *number*, you can specify one or more values from 0 through 7 in binary, decimal or hexadecimal format, as given below: • A single EXP bit—for example, **exp1-except 3** • Several EXP bits—for example, **exp1-except 0,4** • A range of EXP bits—for example, **exp1-except [0-5]**. These values are not supported on filters applied to the loopback interface.
`forwarding-class` *class*	Forwarding class. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`. **NOTE**: On PTX10001-36MR, PTX10003, PTX10004, PTX10008, PTX10016 routers, `exp0` or `exp1` bits are used to obtain the forwarding class.
`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. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.
`interface-set` *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. **NOTE**: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`ip-version` *number*	Match inner IP version. For example, to match MPLS-tagged IPv4 packets, match on the text synonym `ipv4`. Within `ip-version` *number* you can further match packets based on source and destination addresses and ports. Refer Table 58 on page 1011 and Table 59 on page 1012.
`label` *number*	MPLS label value or range of label values in the MPLS header of a packet. For *number*, you can specify one or more values from 0 through 1048575 in decimal or hexadecimal format, as given below: • A single label—for example, **label 3** • Several labels—for example, **label 0,4** • A range of labels—for example, **label [0-5]**. These values are not supported on filters applied to the loopback interface. **NOTE**: This option is deprecated on PTX10001-36MR, PTX10003, PTX10004, PTX10008, and PTX10016 devices and is replaced by `label0`.
`label0` *number*	MPLS label value or range of label values in the TOS MPLS header of a packet. For *number*, you can specify one or more values from 0 through 1048575 in decimal or hexadecimal format, as given below: • A single label—for example, **label0 3** • Several labels—for example, **label0 0,4** • A range of labels—for example, **label0 [0-5]**. These values are not supported on filters applied to the loopback interface.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`label0-except` *number*	Do not match MPLS label value or range of label values in the TOS MPLS header of a packet. For *number*, you can specify one or more values from 0 through 1048575 in decimal or hexadecimal format, as given below: • A single label—for example, **label0-except 3** • Several labels—for example, **label0-except 0,4** • A range of labels—for example, **label0-except [0-5]**. These values are not supported on filters applied to the loopback interface.
`label1` *number*	Match the MPLS label value or range of label values in the MPLS header label of the MPLS header that is next to the TOS MPLS header. For *number*, you can specify one or more values from 0 through 1048575 in decimal or hexadecimal format, as given below: • A single label—for example, **label1 3** • Several labels—for example, **label1 0,4** • A range of labels—for example, **label1 [0-5]**. These values are not supported on filters applied to the loopback interface.
`label1-except` *number*	Do not match on the MPLS label value or range of label values in the MPLS header label of the MPLS header that is next to the TOS MPLS header. For *number*, you can specify one or more values from 0 through 1048575 in decimal or hexadecimal format, as given below: • A single label—for example, **label1-except 3** • Several labels—for example, **label1-except 0,4** • A range of labels—for example, **label1-except [0-5]**. These values are not supported on filters applied to the loopback interface.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description

`label` *number* `top` | `bottom` | `offset` *offset-value*	Match top label, or bottom label or the label at a specified offset (from the top or bottom of the label stack) of the incoming MPLS packet.
	• `top` - Match with reference to top-of-stack towards bottom-of-stack.
	• `bottom` - Match with reference to bottom-of-stack towards top-of-stack.
	• `offset` *<offset-value>* - Match with reference to MPLS stack depth with respect to top or bottom of stack, where *offset-value* = (0..15).
	  • `label` *number* `top offset` *offset-value* - MPLS top label filter match with an offset to stack sanding from 0 to 15. 0 being the first label position from top of stack for both implicit and CLI filters.
	  • `label` *number* `bottom offset` *offset-value* - MPLS bottom label filter match with an offset to stack sanding from 0 to 15. 0 being the first label position from bottom of stack for both implicit and CLI filters.
	  • `label` *number* `offset` *offset-value* - If no options, top or bottom, are provided next to `label` *number* then the default match starts from top-of-stack with given offset. In other words, label number offset [n = 0..15] is equivalent to label number top offset [n = 0..15].
	• `label` *number* - If no options are provided next to `label` *number* then the default match will be done on top label (implicit offset 0 and anchor point being top-of-stack).
	NOTE:
	• Filter match on label with offset out of the MPLS stack depth might not give the expected behaviour.
	  • For filter label match with position as bottom, if offset is out of MPLS stack depth then filter will always match on end-of-stack label.
	  • For filter match with position as top, if offset is out of MPLS stack depth, will point to pay load to match against the configured label.
	NOTE: The configuration command options are introduced in Junos Release 22.3R1.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`loss-priority` *level*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*. **NOTE**: On PTX10001-36MR, PTX10003, PTX10004, PTX10008, PTX10016 routers, `exp0` or `exp1` bits are used to obtain the loss priority.
`loss-priority-except` *level*	Do not match the PLP level. For details, see the `loss-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers.
`source-port` *number*	Match on the TCP or UDP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. In place of the numeric field, you can specify one of the text synonyms listed under `destination-port`.
`ttl0` *number*	Match TTL number or range of numbers in the TOS MPLS header of a packet. Time To Live (TTL) is an 8-bit field in the MPLS label that signifies the remaining time that a packet has left before its life ends and is dropped. For *number*, you can specify a value from 0 through 255.

Table 56: Firewall Filter Match Conditions for MPLS Traffic *(Continued)*

Match Condition	Description
`ttl0-except` *number*	Do not match TTL number or range of numbers in the TOS MPLS header of a packet. Time To Live (TTL) is an 8-bit field in the MPLS label that signifies the remaining time that a packet has left before its life ends and is dropped. For *number*, you can specify a value from 0 through 255.
`ttl1` *number*	Match TTL number or range of numbers in the MPLS header that is next to the TOS MPLS header of a packet. Time To Live (TTL) is an 8-bit field in the MPLS label that signifies the remaining time that a packet has left before its life ends and is dropped. For *number*, you can specify a value from 0 through 255.
`ttl1-except` *number*	Do not match TTL number or range of numbers in the MPLS header that is next to the TOS MPLS header of a packet. Time To Live (TTL) is an 8-bit field in the MPLS label that signifies the remaining time that a packet has left before its life ends and is dropped. For *number*, you can specify a value from 0 through 255.

NOTE: `exp0`, `exp0-except`, `exp1`, `exp1-except`, `ip-version`, `label0`, `label0-except`, `label1`, `label1-except`, `ttl0`, `ttl0-except`, `ttl1`, and `ttl1-except` are only supported on PTX10001-36MR, PTX10003, PTX10004, PTX10008, PTX10016.

Table 57 on page 1008 describes the actions you can configure for MPLS firewall filters at the `[edit firewall family mpls filter` *filter-name* `term` *term-name* `then]` hierarchy level.

Table 57: Supported Actions for MPLS Firewall Filters

Action	Description
`accept`	Accept a packet

Table 57: Supported Actions for MPLS Firewall Filters *(Continued)*

Action	Description
`count` *counter-name*	Count the number of packets that pass this filter or term. **NOTE**: We recommend that you configure a counter for each term in a firewall filter, so that you can monitor the number of packets that match the conditions specified in each filter term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message
`policer`	Starting with Junos OS 13.2X51-D15, you can send traffic matched by an MPLS filter to a two-color policer.
`three-color-policer`	Starting with Junos OS 13.2X51-D15, you can send traffic matched by an MPLS filter to a three-color policer.

RELATED DOCUMENTATION

- Overview of MPLS Firewall Filters on Loopback Interface | **1835**
- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Terminating Actions | **884**
- Firewall Filter Nonterminating Actions | **871**

Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic

IN THIS SECTION

- Matching on IPv4 or IPv6 Packet Header Address or Port Fields in MPLS Flows | **1010**
- IP Address Match Conditions for MPLS Traffic | **1010**
- IP Port Match Conditions for MPLS Traffic | **1011**

Matching on IPv4 or IPv6 Packet Header Address or Port Fields in MPLS Flows

To support network-based service in a core network, you can configure a firewall filter that matches Internet Protocol version 4 (IPv4) or version 6 (IPv6) packet header fields in MPLS traffic (`family mpls`). The firewall filter can match IPv4 or IPv6 packets as an inner payload of an MPLS packet that has a single MPLS label or up to five MPLS labels stacked together. You can configure match conditions based on IPv4 addresses and IPv4 port numbers or IPv6 addresses and IPv6 port numbers in the header.

Firewall filters based on MPLS-tagged IPv4 headers are supported for interfaces on Enhanced Scaling flexible PIC concentrators (FPCs) on T320, T640, T1600, TX Matrix, and TX Matrix Plus routers and switches only. However, the firewall filters based on MPLS-tagged IPv6 headers are supported for interfaces on the Type 5 FPC on T4000 Core Routers only. The feature is not supported for the router or switch loopback interface (`lo0`), the router or switch management interface (`fxp0` or `em0`), or USB modem interfaces (`umd`).

To configure a firewall filter term that matches an address or port fields in the Layer 4 header of packets in an MPLS flow, you use the `ip-version ipv4` match condition to specify that the term is to match packets based on inner IP fields:

- To match an MPLS-tagged IPv4 packet on the source or destination address field in the IPv4 header, specify the match condition at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from ip-version ipv4]` hierarchy level.
- To match an MPLS-tagged IPv4 packet on the source or destination port field in the Layer 4 header, specify the match condition at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from ip-version ipv4 protocol (udp | tcp)]` hierarchy level.

To configure a firewall filter term that matches an address or port fields in the IPv6 header of packets in an MPLS flow, you use the `ip-version ipv6` match condition to specify that the term is to match packets based on inner IP fields:

- To match an MPLS-tagged IPv6 packet on the source or destination address field in the IPv6 header, specify the match condition at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from ip-version ipv6]` hierarchy level.
- To match an MPLS-tagged IPv6 packet on the source or destination port field in the Layer 4 header, specify the match condition at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from ip-version ipv6 protocol (udp | tcp)]` hierarchy level.

IP Address Match Conditions for MPLS Traffic

Table 58 on page 1011 describes the IP address-specific match conditions you can configure at the `[edit firewall family mpls filter` *`filter-name`* `term` *`term-name`* `from ip-version` *`ip-version`*`]` hierarchy level.

Table 58: IP Address-Specific Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
`destination-address` *address*	Match the address of the destination node to receive the packet.
`destination-address` *address* `except`	Do not match the address of the destination node to receive the packet.
`protocol` *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).
`source-address` *address*	Match the address of the source node sending the packet.
`source-address` *address* `except`	Do not match the address of the source node sending the packet.

IP Port Match Conditions for MPLS Traffic

Table 59 on page 1012 describes the IP port-specific *match-conditions* you can configure at the `[edit firewall family mpls filter` *filter-name* `term` *term-name* `from ip-version` *ip-version* `protocol (udp | tcp )]` hierarchy level.

Table 59: IP Port-Specific Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
`destination-port` *number*	Match on the UDP or TCP destination port field. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-port-except` *number*	Do not match on the UDP or TCP destination port field. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` match condition.
`source-port` *number*	Match on the TCP or UDP source port field. In place of the numeric field, you can specify one of the text synonyms listed under `destination-port`.
`source-port-except` *number*	Do not match on the TCP or UDP source port field.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Terminating Actions | **884**
- Firewall Filter Nonterminating Actions | **871**

Firewall Filter Match Conditions for VPLS Traffic

In the `from` statement in the VPLS filter term, you specify conditions that the packet must match for the action in the `then` statement to be taken. All conditions in the `from` statement must match for the action to be taken. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify no match conditions in a term, that term matches all packets.

An individual condition in a `from` statement can contain a list of values. For example, you can specify numeric ranges. You can also specify multiple source addresses or destination addresses. When a condition defines a list of values, a match occurs if one of the values in the list matches the packet.

Individual conditions in a `from` statement can be negated. When you negate a condition, you are defining an explicit mismatch. For example, the negated match condition for `forwarding-class` is `forwarding-class-except`. If a packet matches a negated condition, it is immediately considered not to match the `from` statement, and the next term in the filter is evaluated, if there is one. If there are no more terms, the packet is discarded.

You can configure a firewall filter with match conditions for Virtual Private LAN Service (VPLS) traffic (`family vpls`). Table 60 on page 1013 describes the *`match-conditions`* you can configure at the `[edit firewall family vpls filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

> **NOTE**: Not all match conditions for VPLS traffic are supported on all routing platforms or switching platforms. A number of match conditions for VPLS traffic are supported only on MX Series 5G Universal Routing Platforms.
>
> In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Table 60: Firewall Filter Match Conditions for VPLS Traffic

Match Condition	Description
`destination-mac-address` *`address`*	Match the destination media access control (MAC) address of a VPLS packet.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`destination-port` *number*	(MX Series routers and EX Series switches only) Match the UDP or TCP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-port-except` *number*	(MX Series routers and EX Series switches 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.
`destination-prefix-list` *name*	(ACX Series routers, MX Series routers, and EX Series switches only) Match destination prefixes in the specified list. Specify the name of a prefix list defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level. **NOTE**: VPLS prefix lists support only IPv4 addresses. IPv6 addresses included in a VPLS prefix list will be discarded.
`destination-prefix-list` *name* `except`	(MX Series routers and EX Series switches only) Do not match destination prefixes in the specified list. For more information, see the `destination-prefix-list` match condition.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`dscp` *number*	(MX Series routers and EX Series switches only) Match the 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 Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic. You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: `af11` (10), `af12` (12), `af13` (14), `af21` (18), `af22` (20), `af23` (22), `af31` (26), `af32` (28), `af33` (30), `af41` (34), `af42` (36), `af43` (38)
`dscp-except` *number*	(MX Series routers and EX Series switches only) Do not match on the DSCP. For details, see the `dscp` match condition.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description	
`ether-type` *values*	Match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. You can specify decimal or hexadecimal values from 0 through 65535 (0xFFFF). A value from 0 through 1500 (0x05DC) specifies the length of an Ethernet Version 1 frame. A value from 1536 (0x0600) through 65535 specifies the EtherType (nature of the MAC client protocol) of an Ethernet Version 2 frame. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): `aarp` (0x80F3), `appletalk` (0x809B), `arp` (0x0806), `ipv4` (0x0800), `ipv6` (0x86DD), `mpls-multicast` (0x8848), `mpls-unicast` (0x8847), `oam` (0x8902), `ppp` (0x880B), `pppoe-discovery` (0x8863), `pppoe-session` (0x8864), or `sna` (0x80D5).	
`ether-type-except` *values*	Do not match the 2-octet Length/EtherType field to the specified value or list of values. For details about specifying the *values*, see the `ether-type` match condition.	
`flexible-match-mask` *value*	`bit-length`	Starting in Junos OS 14.2, flexible offset filters are supported in firewall hierarchy configurations. Length of the data to be matched in bits, not needed for string input (0..128)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-mask-name`	Select a flexible match from predefined template field
	`mask-in-hex`	Mask out bits in the packet data to be matched

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description	
	`match-start`	Start point to match in packet
	`prefix`	Value data/string to be matched
`flexible-match-range` *value*	`bit-length`	Length of the data to be matched in bits (0..32)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-range-name`	Select a flexible match from predefined template field
	`match-start`	Start point to match in packet
	`range`	Range of values to be matched
	`range-except`	Do not match this range of values
`forwarding-class` *class*	Match the forwarding class. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`.	
`forwarding-class-except` *class*	Do not match the forwarding class. For details, see the `forwarding-class` match condition.	

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`icmp-code` *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `address-unreachable` (3), `administratively-prohibited` (1), `no-route-to-destination` (0), `port-unreachable` (4)
`icmp-code-except` *message-code*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`icmp-code` *number*	(MX Series routers and EX Series switches only) Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `ip-protocol icmp` or `ip-protocol icmp6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `address-unreachable` (3), `administratively-prohibited` (1), `no-route-to-destination` (0), `port-unreachable` (4)
`icmp-code-except` *number*	(MX Series routers and EX Series switches only) Do not match on the ICMP code field. For details, see the `icmp-code` match condition.
`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. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`interface-group` *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from `0` through `255`. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the `[interfaces` *interface-name* `unit` *number* `family` *family* `filter group]` hierarchy level. For more information, see Filtering Packets Received on a Set of Interface Groups Overview. **NOTE**: This match condition is not supported on T4000 Type 5 FPCs.
`interface-group-except` *group-name*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the `interface-group` match condition. **NOTE**: This match condition is not supported on T4000 Type 5 FPCs.
`interface-set` *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. For more information, see Filtering Packets Received on an Interface Set Overview.
`ip-address` *address*	(MX Series routers and EX Series switches only) 32-bit address that supports the standard syntax for IPv4 addresses. Note that when using this term, the match condition ether-type IPv4 must be defined on the same term.
`ip-destination-address` *address*	(MX Series routers and EX Series switches only) 32-bit address that is the final destination node address for the packet. Note that when using this term, the match condition ether-type IPv4 must be defined on the same term.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ip-precedence` *ip-precedence-field*	(MX Series routers and EX Series switches 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): `critical-ecp` (0xa0), `flash` (0x60), `flash-override` (0x80), `immediate` (0x40), `internet-control` (0xc0), `net-control` (0xe0), `priority` (0x20), or `routine` (0x00).
`ip-precedence-except` *ip-precedence-field*	(MX Series routers and EX Series switches only) Do not match on the IP precedence field.
`ip-protocol` *number*	(MX Series routers and EX Series switches only) IP protocol field.
`ip-protocol-except` *number*	(MX Series routers and EX Series switches only) Do not match on the IP protocol field.
`ip-source-address` *address*	(MX Series routers and EX Series switches only) IP address of the source node sending the packet. Note that when using this term, the match condition ether-type IPv4 must also be defined on the same term.
`ipv6-source-prefix-list` *named-list*	(MX Series only) Match the IPv6 source address in a *named-list*.
`ipv6-address` *address*	(MX Series and EX9200 only) 128-bit address that supports the standard syntax for IPv6 addresses. Starting in Junos OS 14.2, firewall family bridge IPv6 match criteria is supported on MX Series and EX9200 switches.
`ipv6-destination-address` *address*	((MX Series and EX9200 only) 128-bit address that is the final destination node address for this packet. Note that when using this term, the match condition `ether-type IPv6` must be defined on the same term.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-destination-prefix-list` *named-list*	(MX Series only) Match the IPv6 destination addresses in a *named-list*.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-next-header` *protocol*	(MX Series only) Match IPv6 next header protocol type. The following list shows the supported values for *protocol*: • **ah**—IP Security authentication header • **dstopts**—IPv6 destination options • **egp**—Exterior gateway protocol • **esp**—IPSec Encapsulating Security Payload • **fragment**—IPv6 fragment header • **gre**—Generic routing encapsulation • **hop-by-hop**—IPv6 hop by hop options • **icmp**—Internet Control Message Protocol • **icmp6**—Internet Control Message Protocol Version 6 • **igmp**—Internet Group Management Protocol • **ipip**—IP in IP • **ipv6**—IPv6 in IP • **no-next-header**—IPv6 no next header • **ospf**—Open Shortest Path First • **pim**—Protocol Independent Multicast • **routing**—IPv6 routing header • **rsvp**—Resource Reservation Protocol • **sctp**—Stream Control Transmission Protocol • **tcp**—Transmission Control Protocol • **udp**—User Datagram Protocol

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
	• **vrrp**—Virtual Router Redundancy Protocol
`ipv6-next-header-except` *protocol*	(MX Series only) Do not match the IPv6 next header protocol type.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-payload-protocol` *protocol*	(MX Series only) Match IPv6 payload protocol type. The following list shows the supported values for *protocol*: • **ah**—IP Security authentication header • **dstopts**—IPv6 destination options • **egp**—Exterior gateway protocol • **esp**—IPSec Encapsulating Security Payload • **fragment**—IPv6 fragment header • **gre**—Generic routing encapsulation • **hop-by-hop**—IPv6 hop by hop options • **icmp**—Internet Control Message Protocol • **icmp6**—Internet Control Message Protocol Version 6 • **igmp**—Internet Group Management Protocol • **ipip**—IP in IP • **ipv6**—IPv6 in IP • **no-next-header**—IPv6 no next header • **ospf**—Open Shortest Path First • **pim**—Protocol Independent Multicast • **routing**—IPv6 routing header • **rsvp**—Resource Reservation Protocol • **sctp**—Stream Control Transmission Protocol • **tcp**—Transmission Control Protocol • **udp**—User Datagram Protocol

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
	• **vrrp**—Virtual Router Redundancy Protocol
`ipv6-payload-protocol-except` *protocol*	(MX Series only) Do not match the IPv6 payload protocol.
`ipv6-prefix-list` *named-list*	(MX Series only) Match the IPv6 address in a *named-list*.
`ipv6-source-address` *address*	(MX Series only) 128-bit address that is the originating source node address for this packet.
`ipv6-traffic-class` *number*	(MX Series 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 Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic. You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: `af11` (10), `af12` (12), `af13` (14), `af21` (18), `af22` (20), `af23` (22), `af31` (26), `af32` (28), `af33` (30), `af41` (34), `af42` (36), `af43` (38)

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-traffic-class-except` *number*	Do not match the DSCP `number`.
`learn-vlan-1p-priority` *number*	(MX Series routers, M320 router, and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `user-vlan-1p-priority` match condition. **NOTE**: This match condition supports the presence of a control word for MX Series routers and the M320 router.
`learn-vlan-1p-priority-except` *number*	(MX Series routers, M320 router, and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the `learn-vlan-1p-priority` match condition. **NOTE**: This match condition supports the presence of a control word for MX Series routers and the M320 router.
`learn-vlan-dei`	(MX Series routers and EX Series switches only) Match the user VLAN ID drop eligability indicator (DEI) bit.
`learn-vlan-dei-except`	(MX Series routers and EX Series switches only) Do not match the user VLAN ID DEI bit.
`learn-vlan-id` *number*	(MX Series routers and EX Series switches only) VLAN identifier used for MAC learning.
`learn-vlan-id-except` *number*	(MX Series routers and EX Series switches only) Do not match on the VLAN identifier used for MAC learning.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`loss-priority` *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs) and EX Series switches, you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement and about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see Understanding How Forwarding Classes Assign Classes to Output Queues.
`loss-priority-except` *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic.
`port` *number*	(MX Series routers and EX Series switches 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.
`port-except` *number*	(MX Series routers and EX Series switches 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.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`prefix-list` *name*	(MX Series routers and EX Series switches only) Match the destination or source prefixes in the specified list. Specify the name of a prefix list defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level. **NOTE**: VPLS prefix lists support only IPV4 addresses. IPV6 addresses included in a VPLS prefix list will be discarded.
`prefix-list` *name* `except`	(MX Series routers and EX Series switches only) Do not match the destination or source prefixes in the specified list. For more information, see the `destination-prefix-list` match condition.
`source-mac-address` *address*	Source MAC address of a VPLS packet.
`source-port` *number*	(MX Series routers and EX Series switches only) TCP or UDP source port field. You cannot specify the `port` and `source-port` match conditions in the same term.
`source-port-except` *number*	(MX Series routers and EX Series switches 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.
`source-prefix-list` *name*	(ACX Series routers, MX Series routers, and EX Series switches only) Match the source prefixes in the specified prefix list. Specify a prefix list name defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level. **NOTE**: VPLS prefix lists support only IPV4 addresses. IPV6 addresses included in a VPLS prefix list will be discarded.
`source-prefix-list` *name* `except`	(MX Series routers and EX Series switches only) Do not match the source prefixes in the specified prefix list. For more information, see the `source-prefix-list` match condition.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`tcp-flags` *`flags`*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. If you configure this match condition for IPv6 traffic, we recommend that you also configure the `next-header tcp` match condition in the same term to specify that the TCP protocol is being used on the port.
`traffic-type` *`type-name`*	(MX Series routers and EX Series switches only) Traffic type. Specify `broadcast`, `multicast`, `unknown-unicast`, or `known-unicast`.
`traffic-type-except` *`type-name`*	(MX Series routers and EX Series switches only) Do not match on the traffic type. Specify `broadcast`, `multicast`, `unknown-unicast`, or `known-unicast`.

Table 60: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`user-vlan-1p-priority` *number*	(MX Series routers, M320 router, and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `learn-vlan-1p-priority` match condition. **NOTE**: This match condition supports the presence of a control word for MX Series routers and the M320 router.
`user-vlan-1p-priority-except` *number*	(MX Series routers, M320 rouer, and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the `user-vlan-1p-priority` match condition. **NOTE**: This match condition supports the presence of a control word for MX Series routers and the M320 router.
`user-vlan-id` *number*	(MX Series routers and EX Series switches only) Match the first VLAN identifier that is part of the payload.
`user-vlan-id-except` *number*	(MX Series routers and EX Series switches only) Do not match on the first VLAN identifier that is part of the payload.
`vlan-ether-type` *value*	VLAN Ethernet type field of a VPLS packet.
`vlan-ether-type-except` *value*	Do not match on the VLAN Ethernet type field of a VPLS packet.

NOTE: Commit check issues an error if `traffic-type` *known-unicast* or `traffic-type` *unknown-unicast* is unsupported.

Release History Table

Release	Description
14.2	Starting in Junos OS 14.2, flexible offset filters are supported in firewall hierarchy configurations.
14.2	Starting in Junos OS 14.2, firewall family bridge IPv6 match criteria is supported on MX Series and EX9200 switches.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters
- Firewall Filter Terminating Actions
- Firewall Filter Nonterminating Actions

Firewall Filter Match Conditions for Layer 2 CCC Traffic

You can configure a firewall filter with match conditions for Layer 2 circuit cross-connect (CCC) traffic (`family ccc`).

The following restrictions apply to firewall filters for Layer 2 CCC traffic:

- The `input-list` *`filter-names`* and `output-list` *`filter-names`* statements for firewall filters for the `ccc` protocol family are supported on all interfaces with the exception of management interfaces and internal Ethernet interfaces (`fxp` or `em0`), loopback interfaces (`lo0`), and USB modem interfaces (`umd`).
- Only on MX Series routers and EX Series switches, you cannot apply a Layer 2 CCC stateless firewall filter (a firewall filter configured at the `[edit firewall filter family ccc]` hierarchy level) as an output filter. On MX Series routers and EX Series switches, firewall filters configured for the `family ccc` statement can be applied only as input filters.

Table 61 on page 1033 describes the *`match-conditions`* you can configure at the `[edit firewall family ccc filter` *`filter-name`* `term` *`term-name`* `from]` hierarchy level.

Table 61: Firewall Filter Match Conditions for Layer 2 CCC Traffic

Match Condition	Description	
`apply-groups`	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.	
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.	
`destination-mac-address` *address*	(MX Series routers and EX Series switches only) Match the destination media access control (MAC) address of a virtual private LAN service (VPLS) packet. To have packets correctly evaluated by this match condition when applied to egress traffic flowing over a CCC circuit from a logical interface on an I-chip DPC in a Layer 2 virtual private network (VPN) routing instance, you must make a configuration change to the Layer 2 VPN routing instance. You must explicitly disable the use of a control word for traffic flowing out over a Layer 2 circuit. The use of a control word is enabled by default for Layer 2 VPN routing instances to support the emulated virtual circuit (VC) encapsulation for Layer 2 circuits. To explicitly disable the use of a control word for Layer 2 VPNs, include the `no-control-word` statement at either of the following hierarchy levels: • `[edit routing-instances` *routing-instance-name* `protocols l2vpn]` • `[edit logical-systems` *logical-system-name* `routing-instances` *routing-instance-name* `protocols l2vpn]` **NOTE**: This match condition is not supported on PTX series packet transport routers. For more information, see *Disabling the Control Word for Layer 2 VPNs*.	
`flexible-match-mask` *value*	`bit-length`	Length of the data to be matched in bits, not needed for string input (0..128)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)

Table 61: Firewall Filter Match Conditions for Layer 2 CCC Traffic *(Continued)*

Match Condition	Description	
	`byte-offset`	Byte offset after the match start point
	`flexible-mask-name`	Select a flexible match from predefined template field
	`mask-in-hex`	Mask out bits in the packet data to be matched
	`match-start`	Start point to match in packet
	`prefix`	Value data/string to be matched
`flexible-match-range` *value*	`bit-length`	Length of the data to be matched in bits (0..32)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-range-name`	Select a flexible match from predefined template field
	`match-start`	Start point to match in packet
	`range`	Range of values to be matched
	`range-except`	Do not match this range of values

Table 61: Firewall Filter Match Conditions for Layer 2 CCC Traffic *(Continued)*

Match Condition	Description
`forwarding-class` *class*	Forwarding class. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`.
`forwarding-class-except` *class*	Do not match on the forwarding class. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`.
`interface-group` *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from `0` through `255`. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the `[interfaces` *interface-name* `unit` *number* `family` *family* `filter group]` hierarchy level. **NOTE**: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.
`interface-group-except` *number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the `interface-group` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers.
`learn-vlan-1p-priority` *number*	(MX Series routers, M320 router, and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `user-vlan-1p-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers. **NOTE**: This match condition supports the presence of a control word for MX Series and M320 routers.

Table 61: Firewall Filter Match Conditions for Layer 2 CCC Traffic *(Continued)*

Match Condition	Description
`learn-vlan-1p-priority-except` *number*	(MX Series routers, M320 router, and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the `learn-vlan-1p-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers. **NOTE**: This match condition supports the presence of a control word for MX Series and M320 routers.
`loss-priority` *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.
`loss-priority-except` *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. **NOTE**: This match condition is not supported on PTX series packet transport routers. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.

Table 61: Firewall Filter Match Conditions for Layer 2 CCC Traffic *(Continued)*

Match Condition	Description
`user-vlan-1p-priority` *number*	(MX Series routers, M320 router, and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `learn-vlan-1p-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers. **NOTE**: This match condition supports the presence of a control word for MX Series and M320 routers.
`user-vlan-1p-priority-except` *number*	(MX Series routers, M320 router, and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the `user-vlan-1p-priority` match condition. **NOTE**: This match condition is not supported on PTX series packet transport routers. **NOTE**: This match condition supports the presence of a control word for MX Series and M320 routers.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | **814**
- Firewall Filter Terminating Actions | **884**
- Firewall Filter Nonterminating Actions | **871**

Firewall Filter Match Conditions for Layer 2 Bridging Traffic

Only on MX Series routers and EX Series switches, you can configure a standard stateless firewall filter with match conditions for Layer 2 bridging traffic (`family bridge`). Table 62 on page 1038 describes the *match-conditions* you can configure at the `[edit firewall family bridge filter` *filter-name* `term` *term-name* `from]` hierarchy level.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only)

Match Condition	Description
`destination-mac-address` *address*	Destination media access control (MAC) address of a Layer 2 packet in a bridging environment.
`destination-port` *number*	TCP or UDP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term.
`destination-port-except`	Do not match the TCP/UDP destination port.
`destination-prefix-list` *named-list*	Match the IP destination prefixes in a *named-list*.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`dscp` *number*	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 *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*. You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: `af11` (10), `af12` (12), `af13` (14), `af21` (18), `af22` (20), `af23` (22), `af31` (26), `af32` (28), `af33` (30), `af41` (34), `af42` (36), `af43` (38)
`dscp-except` *number*	Do not match on the DSCP number. For more information, see the `dscp-except` match condition.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description	
`ether-type` *value*	Match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. You can specify decimal or hexadecimal values from 0 through 65535 (0xFFFF). A value from 0 through 1500 (0x05DC) specifies the length of an Ethernet Version 1 frame. A value from 1536 (0x0600) through 65535 specifies the EtherType (nature of the MAC client protocol) of an Ethernet Version 2 frame. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): `aarp` (0x80F3), `appletalk` (0x809B), `arp` (0x0806), `ipv4` (0x0800), `ipv6` (0x86DD), `mpls-multicast` (0x8848), `mpls-unicast` (0x8847), `oam` (0x8902), `ppp` (0x880B), `pppoe-discovery` (0x8863), `pppoe-session` (0x8864), `sna` (0x80D5). **NOTE**: When matching on ip-address or ipv6-address, the ether-type ipv4 or ipv6, respectively, must also be specified in order to limit matches to ip traffic only.	
`ether-type-except` *value*	Do not match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. For details about specifying the *values*, see the `ether-type` match condition.	
`flexible-match-mask` *value*	`bit-length`	Length of the data to be matched in bits, not needed for string input (0..128)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-mask-name`	Select a flexible match from predefined template field

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description	
	`mask-in-hex`	Mask out bits in the packet data to be matched
	`match-start`	Start point to match in packet
	`prefix`	Value data/string to be matched
`flexible-match-range` *value*	`bit-length`	Length of the data to be matched in bits (0..32)
	`bit-offset`	Bit offset after the (match-start + byte) offset (0..7)
	`byte-offset`	Byte offset after the match start point
	`flexible-range-name`	Select a flexible match from predefined template field
	`match-start`	Start point to match in packet
	`range`	Range of values to be matched
	`range-except`	Do not match this range of values
`forwarding class` *class*	Forwarding class. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`.	

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`forwarding-class-except` *class*	Ethernet type field of a Layer 2 packet environment. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`.
`icmp-code` *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `ip-protocol icmp`, `ip-protocol icmp6`, or `ip-protocol icmpv6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `address-unreachable` (3), `administratively-prohibited` (1), `no-route-to-destination` (0), `port-unreachable` (4)
`icmp-code-except` *message-code*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`icmp-type` *message-type*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the `ip-protocol icmp`, `ip-protocol icmp6`, or `ip-protocol icmpv6` match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `destination-unreachable` (1), `echo-reply` (129), `echo-request` (128), `membership-query` (130), `membership-report` (131), `membership-termination` (132), `neighbor-advertisement` (136), `neighbor-solicit` (135), `node-information-reply` (140), `node-information-request` (139), `packet-too-big` (2), `parameter-problem` (4), `redirect` (137), `router-advertisement` (134), `router-renumbering` (138), `router-solicit` (133), or `time-exceeded` (3).
`icmp-type-except` *message-type*	Do not match the ICMP message type field. For details, see the `icmp-type` match condition.
`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. **NOTE**: If you configure this match condition with an interface that does not exist, the term does not match any packet.
`interface-group` *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from `0` through `255`. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the `[interfaces` *interface-name* `unit` *number* `family` *family* `filter group]` hierarchy level. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`interface-group-except` *number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the `interface-group` match condition.
`interface-set` *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the `interface-set` statement at the `[edit firewall]` hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.
`ip-address` *address*	32-bit address that supports the standard syntax for IPv4 addresses. **NOTE**: In order to limit matches to IPv4 traffic only, the ether-type ipv4 must also be specified in the same term.
`ip-destination-address` *address*	32-bit address that is the final destination node address for the packet.
`ip-precedence` *ip-precedence-field*	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).
`ip-precedence-except` *ip-precedence-field*	Do not match on the IP precedence field.
`ip-protocol` *number*	IP protocol field.
`ip-protocol-except`	Do not match the IP protocol type.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`ip-source-address` *address*	IP address of the source node sending the packet.
`ipv6-address` *address*	(MX Series only) 128-bit address that supports the standard syntax for IPv6 addresses. **NOTE**: In order to limit matches to IPv6 traffic only, the ether-type ipv6 must also be specified in the same term.
`ipv6-destination-address` *address*	(MX Series only) 128-bit address that is the final destination node address for this packet.
`ipv6-destination-prefix-list` *named-list*	(MX Series only) Match the IPv6 destination addresses in a *named-list*.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`ipv6-next-header` *protocol*	(MX Series only) Match IPv6 next header protocol type. The following list shows the supported values for *protocol*: • **ah**—IP Security authentication header • **dstopts**—IPv6 destination options • **egp**—Exterior gateway protocol • **esp**—IPSec Encapsulating Security Payload • **fragment**—IPv6 fragment header • **gre**—Generic routing encapsulation • **hop-by-hop**—IPv6 hop by hop options • **icmp**—Internet Control Message Protocol • **icmp6**—Internet Control Message Protocol Version 6 • **igmp**—Internet Group Management Protocol • **ipip**—IP in IP • **ipv6**—IPv6 in IP • **no-next-header**—IPv6 no next header • **ospf**—Open Shortest Path First • **pim**—Protocol Independent Multicast • **routing**—IPv6 routing header • **rsvp**—Resource Reservation Protocol • **sctp**—Stream Control Transmission Protocol • **tcp**—Transmission Control Protocol

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
	• **udp**—User Datagram Protocol • **vrrp**—Virtual Router Redundancy Protocol
`ipv6-next-header-except` *protocol*	(MX Series only) Do not match the IPv6 next header protocol type.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`ipv6-payload-protocol` *protocol*	(MX Series only) Match IPv6 payload protocol type. The following list shows the supported values for *protocol*: • **ah**—IP Security authentication header • **dstopts**—IPv6 destination options • **egp**—Exterior gateway protocol • **esp**—IPSec Encapsulating Security Payload • **fragment**—IPv6 fragment header • **gre**—Generic routing encapsulation • **hop-by-hop**—IPv6 hop by hop options • **icmp**—Internet Control Message Protocol • **icmp6**—Internet Control Message Protocol Version 6 • **igmp**—Internet Group Management Protocol • **ipip**—IP in IP • **ipv6**—IPv6 in IP • **no-next-header**—IPv6 no next header • **ospf**—Open Shortest Path First • **pim**—Protocol Independent Multicast • **routing**—IPv6 routing header • **rsvp**—Resource Reservation Protocol • **sctp**—Stream Control Transmission Protocol • **tcp**—Transmission Control Protocol

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
	• **udp**—User Datagram Protocol • **vrrp**—Virtual Router Redundancy Protocol
`ipv6-payload-protocol-except` *protocol*	(MX Series only) Do not match the IPv6 payload protocol.
`ipv6-prefix-list` *named-list*	(MX Series only) Match the IPv6 address in a *named-list*.
`ipv6-source-address` *address*	(MX Series only) 128-bit address that is the originating source node address for this packet.
`ipv6-source-prefix-list` *named-list*	(MX Series only) Match the IPv6 source address in a *named-list*.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`ipv6-traffic-class` *number*	(MX Series 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 *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*. You can specify a numeric value from `0` through `63`. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: `ef` (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: `af11` (10), `af12` (12), `af13` (14), `af21` (18), `af22` (20), `af23` (22), `af31` (26), `af32` (28), `af33` (30), `af41` (34), `af42` (36), `af43` (38)
`ipv6-traffic-class-except` *number*	Do not match the DSCP `number`.
`isid` *number*	(Supported with Provider Backbone Bridging [PBB]) Match internet service identifier.
`isid-dei` *number*	(Supported with PBB) Match the Internet service identifier drop eligibility indicator (DEI) bit.
`isid-dei-except` *number*	(Supported with PBB) Do not match the Internet service identifier DEI bit.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`isid-priority-code-point` *number*	(Supported with PBB) Match the Internet service identifier priority code point.
`isid-priority-code-point-except` *number*	(Supported with PBB) Do not match the Internet service identifier priority code point.
`learn-vlan-1p-priority` *value*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `user-vlan-1p-priority` match condition.
`learn-vlan-1p-priority-except` *value*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the `learn-vlan-1p-priority` match condition.
`learn-vlan-dei` *number*	(Supported with bridging) Match user virtual LAN (VLAN) identifier DEI bit.
`learn-vlan-dei-except` *number*	(Supported with bridging) Do not match user VLAN identifier DEI bit.
`learn-vlan-id` *number*	VLAN identifier used for MAC learning.
`learn-vlan-id-except` *number*	Do not match on the VLAN identifier used for MAC learning.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`loss-priority` *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the `tri-color` statement at the `[edit class-of-service]` hierarchy level to commit a PLP configuration with any of the four levels specified. If the `tri-color` statement is not enabled, you can only configure the `high` and `low` levels. This applies to all protocol families. For information about the `tri-color` statement, see *Configuring and Applying Tricolor Marking Policers*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.
`loss-priority-except` *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*.
`port` *number*	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.
`source-mac-address` *address*	Source MAC address of a Layer 2 packet.
`source-port` *number*	TCP or UDP source port field. You cannot specify the `port` and `source-port` match conditions in the same term.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`source-port-except`	Do not match the TCP/UDP source port.
`tcp-flags` *flags*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. Configuring the `tcp-flags` match condition requires that you configure the `next-header-tcp` match condition.
`traffic-type` *type*	Traffic type. Specify `broadcast`, `multicast`, `unknown-unicast`, or `known-unicast`.
`traffic-type-except` *type*	Do not match on the traffic type.

Table 62: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series Routers and EX Series Switches Only) *(Continued)*

Match Condition	Description
`user-vlan-1p-priority` *value*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from `0` through `7`. Compare with the `learn-vlan-1p-priority` match condition.
`user-vlan-1p-priority-except` *value*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the `user-vlan-1p-priority` match condition.
`user-vlan-id` *number*	(MX Series routers and EX Series switches only) Match the first VLAN identifier that is part of the payload.
`user-vlan-id-except` *number*	(MX Series routers and EX Series switches only) Do not match on the first VLAN identifier that is part of the payload.
`vlan-ether-type` *value*	VLAN Ethernet type field of a Layer 2 bridging packet.
`vlan-ether-type-except` *value*	Do not match on the VLAN Ethernet type field of a Layer 2 bridging packet.

RELATED DOCUMENTATION

- Guidelines for Configuring Firewall Filters | 814
- Firewall Filter Terminating Actions | 884
- Firewall Filter Nonterminating Actions | 871

Firewall Filter Support on Loopback Interface

A loopback interface is a gateway for all the control traffic that enters the Routing Engine of the router. If you want to monitor this control traffic, you must configure a firewall filter on the loopback interface (lo0).

Loopback firewall filters are only applied to packets sent to the Routing Engine for further processing. Both inet and inet6 family filters are supported, and you can apply a firewall filter in the ingress and egress directions on the lo0 interface. However, only `interface-specific` instances of the firewall filter are supported.

For standard firewall filter match conditions, see "Match Conditions for IPv4 Traffic (ACX Series Routers)" on page 917.

The firewall filter on lo0 handles the following exception packets in ingress direction:

- TTL exception packets
- Multicast packets having 224.0.0.x as the destination IP address
- Broadcast packets
- IP option packets

NOTE: Although policer actions can be attached to loopback filters in the ingress direction, the exact behavior depends on the CPU RX queue configurations. For example, rate limiting in ingress direction (through policer configuration) occurs after any CPU rate limiters.

The following is a sample configuration for attaching a firewall to the loopback interface:

```
[edit interfaces]
lo0 {
    unit 0 {
        family <inet | inet6> {
            filter {
                input f1;
            }
        }
    }
}
family <inet | inet6>{
    filter f1 {
        interface-specific; >> Mandatory Field.
```

```
        term t1 {
            from {
                protocol ospf;
            }
            then {
                count c1;
                discard;
            }
        }
        term t2 {
            then {
                count c2;
                accept;
            }
        }
    }
}
```

CHAPTER 15

Applying Firewall Filters to Routing Engine Traffic

IN THIS CHAPTER

- Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs | **1057**
- Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List | **1059**
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | **1065**
- Example: Configure a Filter to Block Telnet and SSH Access | **1073**
- Example: Configuring a Filter to Block TFTP Access | **1086**
- Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags | **1091**
- Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers | **1095**
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods | **1103**
- Example: Protecting the Routing Engine with a Packets-Per-Second Rate Limiting Filter | **1119**
- Example: Configuring a Filter to Exclude DHCPv6 and ICMPv6 Control Traffic for LAC Subscriber | **1124**
- Port Number Requirements for DHCP Firewall Filters | **1131**
- Example: Configuring a DHCP Firewall Filter to Protect the Routing Engine | **1132**

Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs

For Layer 3 VPNs (VRF routing instances), you can configure a logical unit on the loopback interface into each VRF routing instance that you have configured on the router. Associating a VRF routing instance with a logical unit on the loopback interface allows you to easily identify the VRF routing instance.

Doing this is useful for troubleshooting:

- It allows you to ping a remote CE router from a local PE router in a Layer 3 VPN. For more information, see *Example: Troubleshooting Layer 3 VPNs*.
- It ensures that a path maximum transmission unit (MTU) check on traffic originating on a VRF or virtual-router routing instance functions properly. For more information, see *Configuring Path MTU Checks for VPN Routing Instances*.

You can also configure a firewall filter for the logical unit on the loopback interface; this configuration allows you to filter traffic for the VRF routing instance associated with it.

The following describes how firewall filters affect the VRF routing instance depending on whether they are configured on the default loopback interface, the VRF routing instance, or some combination of the two. The "default loopback interface" refers to `lo0.0` (associated with the default routing table), and the "VRF loopback interface" refers to `lo0.n`, which is configured in the VRF routing instance.

- If you configure Filter A on the default loopback interface and Filter B on the VRF loopback interface, the VRF routing instance uses Filter B.
- If you configure Filter A on the default loopback interface but do not configure a filter on the VRF loopback interface, the VRF routing instance does not use a filter.
- If you configure Filter A on the default loopback interface but do not configure a VRF loopback interface, the VRF routing instance uses Filter A. For MX80 devices, the behavior is slightly different: If you configure filters on the default loopback interface but do not configure a VRF loopback interface, the VRF routing instance uses only the input filters assigned to the default loopback (it does not use output filters from the default loopback).

For some ACX Series Universal Metro Routers (ACX1000, ACX2000, ACX4000, and ACX5000), the default loopback filter must be in the same routing, or virtual routing and forwarding (VRF), instance as the ingress traffic it filters. That is, on these devices, the default loopback filter cannot be used for traffic traversing an interface that belongs to a different routing instance.

To configure a logical unit on the loopback interface, include the `unit` statement:

```
unit number {
    family inet {
        address address;
    }
}
```

You can include this statement at the following hierarchy levels:

- `[edit interfaces lo0]`
- `[edit logical-systems logical-system-name interfaces lo0]`

To associate a firewall filter with the logical unit on the loopback interface, include the `filter` statement:

```
filter {
    input filter-name;
}
```

You can include this statement at the following hierarchy levels:

- [edit interfaces lo0 unit *unit-number* family inet]
- [edit logical-systems *logical-system-name* interfaces lo0 unit *unit-number* family inet]

To include the lo0.*n* interface (where *n* specifies the logical unit) in the configuration for the VRF routing instance, include the following statement:

```
interface lo0.n;
```

You can include this statement at the following hierarchy levels:

- [edit routing-instances *routing-instance-name*]
- [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]

Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List

IN THIS SECTION

- Requirements | 1059
- Overview | 1060
- Configuration | 1060
- Verification | 1063

This example shows how to configure a standard stateless firewall filter that limits certain TCP and Internet Control Message Protocol (ICMP) traffic destined for the Routing Engine by specifying a list of prefix sources that contain allowed BGP peers.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1060

In this example, you create a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except BGP peers that have a specified prefix.

Topology

A source prefix list, **plist_bgp179**, is created that specifies the list of source prefixes that contain allowed BGP peers.

The stateless firewall filter **filter_bgp179** matches all packets from the source prefix list **plist_bgp179** to the destination port number 179.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1060
- Configure the Filter | 1061
- Results | 1062

CLI Quick Configuration

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

```
set policy-options prefix-list plist_bgp179 apply-path "protocols bgp group <*> neighbor <*>"
set firewall family inet filter filter_bgp179 term 1 from source-address 0.0.0.0/0
set firewall family inet filter filter_bgp179 term 1 from source-prefix-list plist_bgp179 except
set firewall family inet filter filter_bgp179 term 1 from destination-port bgp
set firewall family inet filter filter_bgp179 term 1 then reject
```

```
set firewall family inet filter filter_bgp179 term 2 then accept
set interfaces lo0 unit 0 family inet filter input filter_bgp179
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configure the Filter

Step-by-Step Procedure

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

To configure the filter:

1. Expand the prefix list **bgp179** to include all prefixes pointed to by the BGP peer group defined by **protocols bgp group <*> neighbor <*>**.

```
[edit policy-options prefix-list plist_bgp179]
user@host# set apply-path " protocols bgp group <*> neighbor <*>"
```

2. Define the filter term that rejects TCP connection attempts to port 179 from all requesters except the specified BGP peers.

```
[edit firewall family inet filter filter_bgp179]
user@host# set term term1 from source-address 0.0.0.0/0
user@host# set term term1 from source-prefix-list bgp179 except
user@host# set term term1 from destination-port bgp
user@host# set term term1 then reject
```

3. Define the other filter term to accept all packets.

```
[edit firewall family inet filter filter_bgp179]
user@host# set term term2 then accept
```

4. Apply the firewall filter to the loopback interface.

```
[edit interfaces lo0 unit 0 family inet]
user@host# set filter input filter_bgp179
user@host# set address 127.0.0.1/32
```

Results

From configuration mode, confirm your configuration by entering the **show firewall**, **show interfaces**, and **show policy-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
    filter filter_bgp179 {
        term 1 {
            from {
                source-address {
                    0.0.0.0/0;
                }
                source-prefix-list {
                    plist_bgp179 except;
                }
                destination-port bgp;
            }
            then {
                reject;
            }
        }
        term 2 {
            then {
                accept;
            }
        }
    }
}
```

```
user@host# show interfaces
lo0 {
```

```
    unit 0 {
        family inet {
            filter {
                input filter_bgp179;
            }
            address 127.0.0.1/32;
        }
    }
}
```

```
user@host# show policy-options
prefix-list plist_bgp179 {
    apply-path "protocols bgp group <*> neighbor <*>";
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filter Applied to the Loopback Interface | 1063

Confirm that the configuration is working properly.

Displaying the Firewall Filter Applied to the Loopback Interface

Purpose

Verify that the firewall filter **filter_bgp179** is applied to the IPv4 input traffic at logical interface **lo0.0**.

Action

Use the `show interfaces statistics operational mode` command for logical interface **lo0.0**, and include the **detail** option. Under the **Protocol inet** section of the command output section, the **Input Filters** field displays the name of the stateless firewall filter applied to the logical interface in the input direction.

```
[edit]
user@host> show interfaces statistics lo0.0 detail
  Logical interface lo0.0 (Index 321) (SNMP ifIndex 16) (Generation 130)
    Flags: SNMP-Traps Encapsulation: Unspecified
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                   0 bps
     Output bytes  :                   0                   0 bps
     Input  packets:                   0                   0 pps
     Output packets:                   0                   0 pps
    Protocol inet, MTU: Unlimited, Generation: 145, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter_bgp179
      Addresses, Flags: Primary
        Destination: Unspecified, Local: 127.0.0.1, Broadcast: Unspecified, Generation: 138
```

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | **778**

Firewall Filter Match Conditions Based on Address Fields | **982**

Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods | **1103**

Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags | **1091**

prefix-list

Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources

IN THIS SECTION

- Requirements | 1065
- Overview | 1065
- Configuration | 1066
- Verification | 1069

This example shows how to create a stateless firewall filter that protects the Routing Engine from traffic originating from untrusted sources.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

In this example, you create a stateless firewall filter called protect-RE that discards all traffic destined for the Routing Engine except SSH and BGP protocol packets from specified trusted sources. This example includes the following firewall filter terms:

- `ssh-term`—Accepts TCP packets with a source address of `192.168.122.0/24` and a destination port that specifies SSH.
- `bgp-term`—Accepts TCP packets with a source address of `10.2.1.0/24` and a destination port that specifies BGP.
- `discard-rest-term`—For all packets that are not accepted by `ssh-term` or `bgp-term`, creates a firewall filter log and system logging records, then discards all packets.

NOTE: You can move terms within the firewall filter using the `insert` command. See *insert* in the Junos OS CLI User Guide.

Configuration

IN THIS SECTION

- Procedure | 1066

Procedure

CLI Quick Configuration

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

```
set firewall family inet filter protect-RE term ssh-term from source-address 192.168.122.0/24
set firewall family inet filter protect-RE term ssh-term from protocol tcp
set firewall family inet filter protect-RE term ssh-term from destination-port ssh
set firewall family inet filter protect-RE term ssh-term then accept
set firewall family inet filter protect-RE term bgp-term from source-address 10.2.1.0/24
set firewall family inet filter protect-RE term bgp-term from protocol tcp
set firewall family inet filter protect-RE term bgp-term from destination-port bgp
set firewall family inet filter protect-RE term bgp-term then accept
set firewall family inet filter protect-RE term discard-rest-term then log
set firewall family inet filter protect-RE term discard-rest-term then syslog
set firewall family inet filter protect-RE term discard-rest-term then discard
set interfaces lo0 unit 0 family inet filter input protect-RE
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure the stateless firewall filter:

1. Create the stateless firewall filter.

```
[edit]
user@host# edit firewall family inet filter protect-RE
```

2. Create the first filter term.

```
[edit firewall family inet filter protect-RE]
user@host# edit term ssh-term
```

3. Define the protocol, destination port, and source address match conditions for the term.

```
[edit firewall family inet filter protect-RE term ssh-term]
user@host# set from protocol tcp destination-port ssh source-address 192.168.122.0/24
```

4. Define the actions for the term.

```
[edit firewall family inet filter protect-RE term ssh-term]
user@host# set then accept
```

5. Create the second filter term.

```
[edit firewall family inet filter protect-RE]
user@host# edit term bgp-term
```

6. Define the protocol, destination port, and source address match conditions for the term.

```
[edit firewall family inet filter protect-RE term bgp-term]
user@host# set from protocol tcp destination-port bgp source-address 10.2.1.0/24
```

7. Define the action for the term.

```
[edit firewall family inet filter protect-RE term bgp-term]
user@host# set then accept
```

8. Create the third filter term.

```
[edit firewall family inet filter protect-RE]
user@host# edit term discard-rest-term
```

9. Define the action for the term.

```
[edit firewall family inet filter protect-RE term discard-rest]
user@host# set then log syslog discard
```

10. Apply the filter to the input side of the Routing Engine interface.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet filter input protect-RE
```

Results

Confirm your configuration by entering the `show firewall` command and the `show interfaces lo0` command from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
    filter protect-RE {
        term ssh-term {
            from {
                source-address {
                    192.168.122.0/24;
                }
                protocol tcp;
                destination-port ssh;
            }
            then accept;
        }
        term bgp-term {
            from {
                source-address {
                    10.2.1.0/24;
                }
```

```
                protocol tcp;
                destination-port bgp;
            }
            then accept;
        }
        term discard-rest-term {
            then {
                log;
                syslog;
                discard;
            }
        }
    }
}
```

```
user@host# show interfaces lo0
unit 0 {
    family inet {
        filter {
            input protect-RE;
        }
        address 127.0.0.1/32;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

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

Verification

IN THIS SECTION

- Displaying Stateless Firewall Filter Configurations | **1070**
- Verifying a Services, Protocols, and Trusted Sources Firewall Filter | **1070**
- Displaying Stateless Firewall Filter Logs | **1071**

To confirm that the configuration is working properly, perform these tasks:

Displaying Stateless Firewall Filter Configurations

Purpose

Verify the configuration of the firewall filter.

Action

From configuration mode, enter the `show firewall` command and the `show interfaces lo0` command.

Meaning

Verify that the output shows the intended configuration of the firewall filter. In addition, verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the `insert` CLI command.

Verifying a Services, Protocols, and Trusted Sources Firewall Filter

Purpose

Verify that the actions of the firewall filter terms are taken.

Action

Send packets to the device that match the terms. In addition, verify that the filter actions are *not* taken for packets that do not match.

- Use the `ssh` *`host-name`* command from a host at an IP address that matches `192.168.122.0/24` to verify that you can log in to the device using only SSH from a host with this address prefix.
- Use the `show route summary` command to verify that the routing table on the device does not contain any entries with a protocol other than `Direct`, `Local`, `BGP`, or `Static`.

Sample Output

command-name

```
% ssh 192.168.249.71
%ssh host
```

```
user@host's password:
--- JUNOS 6.4-20040518.0 (JSERIES) #0: 2004-05-18 09:27:50 UTC

user@host>
```

command-name

```
user@host> show route summary
Router ID: 192.168.249.71

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
              Direct:     10 routes,      9 active
               Local:      9 routes,      9 active
                 BGP:     10 routes,     10 active
              Static:      5 routes,      5 active
...
```

Meaning

Verify the following information:

- You can successfully log in to the device using SSH.
- The `show route summary` command does not display a protocol other than `Direct`, `Local`, `BGP`, or `Static`.

Displaying Stateless Firewall Filter Logs

Purpose

Verify that packets are being logged. If you included the `log` or `syslog` action in a term, verify that packets matching the term are recorded in the firewall log or your system logging facility.

Action

From operational mode, enter the `show firewall log` command.

Sample Output

command-name

```
user@host> show firewall log
Log :
Time      Filter    Action Interface     Protocol Src Addr      Dest Addr
15:11:02  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
...
```

Meaning

Each record of the output contains information about the logged packet. Verify the following information:

- Under `Time`, the time of day the packet was filtered is shown.
- The `Filter` output is always `pfe`.
- Under `Action`, the configured action of the term matches the action taken on the packet—`A` (accept), `D` (discard), `R` (reject).
- Under `Interface`, the inbound (ingress) interface on which the packet arrived is appropriate for the filter.
- Under `Protocol`, the protocol in the IP header of the packet is appropriate for the filter.
- Under `Src Addr`, the source address in the IP header of the packet is appropriate for the filter.
- Under `Dest Addr`, the destination address in the IP header of the packet is appropriate for the filter.

RELATED DOCUMENTATION

show route summary

show firewall

show firewall log

show interfaces (Loopback)

Example: Configure a Filter to Block Telnet and SSH Access

IN THIS SECTION

- Requirements | 1073
- Overview and Topology | 1073
- Configuration | 1074
- Verify the Stateless Firewall Filter | 1082

Requirements

You need two devices running Junos OS with a shared network link. No special configuration beyond basic device initialization (management interface, remote access, user login accounts, etc.) is required before configuring this example. While not a strict requirement, console access to the R2 device is recommended.

NOTE: Our content testing team has validated and updated this example.

Overview and Topology

IN THIS SECTION

- Example Topology | 1074

In this example, you create an IPv4 stateless firewall filter that logs and rejects Telnet or SSH packets sent to the local Routing Engine, unless the packet originates from the 192.168.1.0/30 subnet. The filter is applied to the loopback interface to ensure that only traffic destined to the local device is affected. You apply the filter in the input direction. An output filter is not used. As a result all locally generated traffic is allowed.

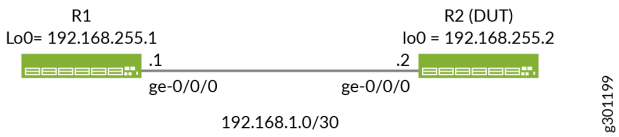
- To match packets originating from a specific subnet or IP prefix, you use the `source-address` IPv4 match condition applied in the input direction.

- To match packets destined for the Telnet port and SSH ports, you use the `protocol tcp` match condition combined with a `port telnet` and `port ssh` IPv4 match conditions applied in the input direction.

Example Topology

Figure 51 on page 1074 shows the test topology for this example. The firewall filter is applied to the R2 device, making it the device under test (DUT). The R1 and the R2 devices share a link that is assigned a subnet of 192.168.1.0/30. Both devices have loopback addresses assigned from the 192.168.255.0/30 prefix using a /32 subnet mask. Static routes provide reachability between loopback addresses because an interior gateway protocol is not configured in this basic example.

Figure 51: Example Topology



Configuration

IN THIS SECTION

- CLI Quick Configuration | **1075**
- Configure the R1 Device | **1076**
- Verify and Commit the Configuration at the R1 Device | **1077**
- Configure the R2 Device | **1078**
- Verify and Commit the Configuration at Device R2 | **1080**

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

CAUTION: By design the sample filter restricts Telnet and SSH access to R2 unless it originates from the shared subnet at R1. If you use SSH or Telnet to access the R2 device directly, you will lose connectivity when the filter is applied. We recommend that

you have console access when configuring this example. If needed you can use the R1 device as a jump host to launch an SSH session to R2 after the filter is applied. Alternatively, consider modifying the sample filter to also permit the IP subnet assigned to the machine you use to access the R2 device.

Perform the following tasks to configure this example:

CLI Quick Configuration

Quick Configuration for the R1 Device

To quickly configure the R1 device, edit the following commands as needed and paste them into the CLI at the `[edit]` hierarchy level. Be sure to issue a `commit`in configuration mode to activate the changes.

```
set system host-name R1
set system services ssh root-login allow
set interfaces ge-0/0/0 description "Link from R1 to R2"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
set interfaces lo0 unit 0 family inet address 192.168.255.1/32
set routing-options static route 192.168.255.2/32 next-hop 192.168.1.2
```

Quick Configuration for the R2 Device

To quickly configure the R2 device, edit the following commands as needed and paste them into the CLI at the `[edit]` hierarchy level. Be sure to issue a `commit` in configuration mode to activate the changes.

TIP: Consider using `commit-confirmed` when making changes that might affect remote access to your device. *Activating a Junos OS Configuration but Requiring Confirmation*

```
set system host-name R2
set system services ssh root-login allow
set system services telnet
set interfaces ge-0/0/0 description "Link from R2 to R1"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/30
set interfaces lo0 unit 0 family inet filter input local_acl
set interfaces lo0 unit 0 family inet address 192.168.255.2/32
set firewall family inet filter local_acl term terminal_access from source-address 192.168.1.0/30
set firewall family inet filter local_acl term terminal_access from protocol tcp
set firewall family inet filter local_acl term terminal_access from port ssh
```

```
set firewall family inet filter local_acl term terminal_access from port telnet
set firewall family inet filter local_acl term terminal_access then accept
set firewall family inet filter local_acl term terminal_access_denied from protocol tcp

set firewall family inet filter local_acl term tcp-estab from protocol tcp
set firewall family inet filter local_acl term tcp-estab from tcp-established
set firewall family inet filter local_acl term tcp-estab then accept
set firewall family inet filter local_acl term terminal_access_denied from port ssh
set firewall family inet filter local_acl term terminal_access_denied from port telnet
set firewall family inet filter local_acl term terminal_access_denied then log
set firewall family inet filter local_acl term terminal_access_denied then reject
set firewall family inet filter local_acl term default-term then accept
set routing-options static route 192.168.255.1/32 next-hop 192.168.1.1
```

Configure the R1 Device

Step-by-Step Procedure

Follow these steps to configure the R1 device:

1. Configure the interfaces:

```
[edit]
user@R1# set interfaces ge-0/0/0 description "Link from R1 to R2"
user@R1# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
user@R1# set interfaces lo0 unit 0 family inet address 192.168.255.1/32
```

2. Configure the host name and static route to the R2 device's loopback address. You also configure Telnet and SSH access:

```
[edit]
user@R1# set system host-name R1
user@R1# set system services ssh root-login allow
user@R1# set system services telnet
user@R1# set routing-options static route 192.168.255.2/32 next-hop 192.168.1.2
```

Verify and Commit the Configuration at the R1 Device

Step-by-Step Procedure

Complete the following steps to verify and commit your candidate configuration at the R1 device:

1. Confirm interface configuration with the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R1# show interfaces
ge-0/0/0 {
    description "Link from R1 to R2";
    unit 0 {
        family inet {
            address 192.168.1.1/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.255.1/32;
        }
    }
}
```

2. Verify the static route used to reach the R2 device's loopback address and that SSH and Telnet access are enabled. Use the `show routing-options` and `show system services` configuration mode commands. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R1# show routing-options
static {
    route 192.168.255.2/32 next-hop 192.168.1.2;
}
user@R1# show system services
ssh {
    root-login allow;
```

```
    }
telnet;
```

3. When satisfied with the configuration on the R1 device, commit your candidate configuration.

```
[edit]
user@R1# commit
```

Configure the R2 Device

Step-by-Step Procedure

Complete the following steps to configure the R2 device. You begin by defining the stateless firewall filter that selectively blocks Telnet and SSH access:

1. Position yourself at the `edit firewall family inet filter` *local_acl* hierarchy:

```
[edit]
user@R2# edit firewall family inet filter local_acl
```

2. Define the filter term *terminal_access*. This term permits Telnet and SSH from the specified source prefix(s):

```
[edit firewall family inet filter local_acl]
user@R2# set term terminal_access from source-address 192.168.1.0/30
user@R2# set term terminal_access from protocol tcp
user@R2# set term terminal_access from port ssh
user@R2# set term terminal_access from port telnet
user@R2# set term terminal_access then accept
```

3. Define the filter term *terminal_access_denied*. This term rejects SSH and Telnet from *all other* source addresses. This term is configured to log matches to the term and to generate an explicit Internet Control Message Protocol (ICMP) destination unreachable response back to the packet's source. See "Firewall Filter Logging Actions" on page 1282 for details on filter logging options.

TIP: You can use the `discard` action to suppress generation of ICMP error messages back to the source. See "Firewall Filter Terminating Actions" on page 884 for details.

```
[edit firewall family inet filter local_acl]
user@R2# set term terminal_access_denied from protocol tcp
user@R2# set term terminal_access_denied from port ssh
user@R2# set term terminal_access_denied from port telnet
user@R2# set term terminal_access_denied then log
user@R2# set term terminal_access_denied then reject
user@R2# set term default-term then accept
```

4. Optional.

   Define the filter term *tcp-estab*. This term permits outbound access to the Internet to support connections to the Juniper Mist cloud (*tcp-established* is a bit-field match condition, **tcp-flags "(ack | rst)"**, which indicates an established TCP session, but not the first packet of a TCP connection):

```
[edit firewall family inet filter local_acl]
user@R2# set term tcp-estab from protocol tcp
user@R2# set term tcp-estab from tcp-established
user@R2# set term tcp-estab then accept
```

5. Define the filter term *default-term*. This term accepts all other traffic. Recall that Junos OS stateless filters have an implicit *deny* term at their end. The *default-term* overrides this behavior by terminating the filter with an explicit *accept* action. The termination of the filter results in all other traffic being accepted by the filer.

   NOTE: For this example we are allowing all other traffic, but for your network you might want to secure the routing engine. See protecting the routing engine for more information.

```
[edit firewall family inet filter local_acl]
user@R2# set term default-term then accept
```

6. Configure the loopback interface, and apply the filter in the input direction:

```
[edit]
user@R2# set interfaces lo0 unit 0 family inet filter input local_acl
user@R2# set interfaces lo0 unit 0 family inet address 192.168.255.2/32
```

7. Configure the host name, the ge-0/0/0 interface, the static route to the R1 device's loopback address, and enable remote access through SSH and Telnet:

```
[edit]
user@R2# set system host-name R2
user@R2# set system services ssh root-login allow
user@R2# set system services telnet
user@R2# set interfaces ge-0/0/0 description "Link from R2 to R1"
user@R2# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.2/30
user@R2# set routing-options static route 192.168.255.1/32 next-hop 192.168.1.1
```

Verify and Commit the Configuration at Device R2

Step-by-Step Procedure

Complete the following steps to verify and commit your candidate configuration at the R2 device:

1. Confirm the configuration of the stateless firewall filter with the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R2# show firewall
family inet {
    filter local_acl {
        term terminal_access {
            from {
                source-address {
                    192.168.1.0/30;
                }
                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;
        }
    }
}
```

2. Confirm interface configuration and filter application with the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R2# show interfaces
ge-0/0/0 {
    description "Link from R2 to R1";
    unit 0 {
        family inet {
            address 192.168.1.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            filter {
                input local_acl;
            }
            address 192.168.255.2/32;
        }
    }
}
```

3. Verify the static route used to reach the loopback address of the R1 device, and verify that Telnet and SSH access are enabled. Use the `show routing-options` and `show system services` configuration mode commands. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R2# show routing-options
static {
    route 192.168.255.1/32 next-hop 192.168.1.1;
}
user@R2# show system services
ssh {
    root-login allow;
    }
telnet;
```

4. When satisfied with the configuration on the R2 device, commit your candidate configuration.

TIP: Consider using `commit-confirmed` when making changes that might affect remote access to your device.

```
[edit]
user@R2# commit
```

Verify the Stateless Firewall Filter

IN THIS SECTION

- Verify Accepted Packets | **1083**
- Verify Logged and Rejected Packets | **1084**

Confirm that the firewall filter to limit Telnet and SSH access is working properly.

Verify Accepted Packets

Purpose

Verify that the firewall filter correctly allows SSH and Telnet when the traffic is sourced from the 192.168.1.0/30 subnet.

Action

1. Clear the firewall log on your router or switch.

```
user@R2> clear firewall log
```

2. From a host at an IP address *within* the 192.168.1.0/30 subnet, use a `ssh` *192.168.255.2* command to verify that you can log in to the device using SSH from an allowed source address. This packet should be accepted, but the packet header information for this packet should not be logged in the firewall filter log buffer in the Packet Forwarding Engine. You will be prompted to save the SSH host key if this is the first SSH login as *user* between these devices.

> **NOTE**: By default the R1 device will source the SSH traffic from the egress interface used to reach the destination. As a result this traffic is sourced from the 192.168.1.1 address assigned to the R1 device's ge-0/0/0 interface.

```
user@R1>ssh 192.168.255.2
Password:
Last login: Wed Aug 19 09:23:58 2020 from 192.168.1.1
--- JUNOS 20.2R1.10 Kernel 64-bit  JNPR-11.0-20200608.0016468_buil
user@R2>
```

3. Log out of the CLI at the R2 device to close the SSH session.

```
user@R2> exit
logout
Connection to 192.168.255.2 closed.
user@R1>
```

4. From a host at an IP address *within* the 192.168.1.0/30 subnet, use the `telnet` *192.168.255.2* command to verify that you can log in to your router or switch using Telnet from an allowed source address.

This packet should be accepted, but the packet header information for this packet should not be logged in the firewall filter log buffer in the Packet Forwarding Engine.

```
user@host-A> telnet 192.168.255.2
Trying 192.168.255.2...
Connected to 192.168.255.2.
Escape character is '^]'.
login: user
Password:

--- JUNOS 20.2R1.10 Kernel 64-bit  JNPR-11.0-20200608.0016468_buil
user@R2>
```

5. Log out of the CLI to close the Telnet session to the R2 device.

```
user@R2:~ # exit
Connection closed by foreign host.

root@R1>
```

6. Use the `show firewall log` command to verify that the firewall log buffer on the R2 device's Packet Forwarding Engine (PFE) *does not* contain any entries with a source address in the 192.168.1.0/30 subnet.

```
user@R2> show firewall log
```

Verify Logged and Rejected Packets

Purpose

Verify that the firewall filter correctly rejects SSH and Telnet traffic that does *not* originate from the 192.168.1.0/30 subnet.

Action

1. Clear the firewall log on your router or switch.

```
user@R2> clear firewall log
```

2. Generate SSH traffic sourced from the loopback address of the R1 device. The source address of this traffic is *outside of* the allowed 192.168.1.0/30 subnet. Use the `ssh` *192.168.255.2* `source` *192.168.255.1* command to verify that you *cannot* log in to the device using SSH from this source address. This packet should be rejected, and the packet header information should be logged in the firewall filter log buffer.

```
user@R1 ssh 192.168.255.2 source 192.168.255.1
ssh: connect to host 192.168.255.2 port 22: Connection refused

root@R1>
```

   The output shows that the SSH connection is rejected. This output confirms that the filter is generating an ICMP error message and that it correctly blocks SSH traffic when sent from a disallowed source address.

3. Generate Telnet traffic sourced from the loopback address of the R1 device. The source address of this traffic is *outside of* the allowed 192.168.1.0/30 subnet. Use the `telnet` *192.168.255.2* `source` *192.168.255.1* command to verify that you *cannot* log in to the device using Telnet from this source address. This packet should be rejected, and the packet header information for this packet should be logged in the firewall filter log buffer in the PFE.

```
user@R1> telnet 192.168.255.2 source 192.168.255.1
Trying 192.168.255.2...
telnet: connect to address 192.168.255.2: Connection refused
telnet: Unable to connect to remote host
```

   The output shows that the Telnet connection is rejected. This output confirms that the filter is generating an ICMP error message and that it correctly blocks Telnet traffic when sent from a disallowed source address.

4. Use the `show firewall log` command to verify that the firewall log buffer on the R2 device contains entries showing that packets with a source address of 192.168.255.1 were rejected.

```
user@R2> show firewall log
Log :
Time      Filter  Action  Interface     Protocol   Src Addr        Dest Addr
```

```
15:17:11  pfe     R       ge-0/0/0.0    TCP        192.168.255.1   192.168.255.2
15:12:04  pfe     R       ge-0/0/0.0    TCP        192.168.255.1   192.168.255.2
```

The output confirms that traffic from the 192.168.255.1 source address matched the filter's *terminal_access_denied* term. The `Action` column displays an `R` to indicate that these packets were rejected. The interface, transport protocol, and source and destination addresses are also listed. These results confirm that the firewall filter is working properly for this example.

Example: Configuring a Filter to Block TFTP Access

IN THIS SECTION

- Requirements | **1086**
- Overview | **1086**
- Configuration | **1087**
- Verification | **1090**

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | **1087**

By default, to decrease vulnerability to denial-of-service (DoS) attacks, the Junos OS 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.

Topology

To interoperate with other vendors' equipment, you can configure a filter that checks for both of these addresses and overrides the default RPF-check filter by accepting these packets. In this example, you block Trivial File Transfer Protocol (TFTP) access, logging any attempts to establish TFTP connections.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1087**
- Configure the Stateless Firewall Filter | **1087**
- Apply the Firewall Filter to the Loopback Interface | **1088**
- Confirm and Commit Your Candidate Configuration | **1088**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter tftp_access_control term one from protocol udp
set firewall family inet filter tftp_access_control term one from port tftp
set firewall family inet filter tftp_access_control term one then log
set firewall family inet filter tftp_access_control term one then discard
set interfaces lo0 unit 0 family inet filter input tftp_access_control
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter that selectively blocks TFTP access:

1. Create the stateless firewall filter `tftp_access_control`.

```
[edit]
user@host# edit firewall family inet filter tftp_access_control
```

2. Specify a match on packets received on UDP port 69.

```
[edit firewall family inet filter tftp_access_control]
user@host# set term one from protocol udp
user@host# set term one from port tftp
```

3. Specify that matched packets be logged to the buffer on the Packet Forwarding Engine and then discarded.

```
[edit firewall family inet filter tftp_access_control]
user@host# set term one then log
user@host# set term one then discard
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

-
```
[edit]
user@host# set interfaces lo0 unit 0 family inet filter input tftp_access_control
user@host# set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter tftp_access_control {
        term one {
            from {
                protocol udp;
                port tftp;
            }
            then {
                log;
                discard;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
                input tftp_access_control;
            }
            address 127.0.0.1/32;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

IN THIS SECTION

- Verifying Logged and Discarded Packets | 1090

Confirm that the configuration is operating properly:

Verifying Logged and Discarded Packets

Purpose

Verify that the actions of the firewall filter terms are taken.

Action

To

1. Clear the firewall log on your router or switch.

```
user@myhost> clear firewall log
```

2. From another host, send a packet to UDP port 69 on this router or switch.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | 778
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | 1065
- Example: Configure a Filter to Block Telnet and SSH Access | 1073
- Example: Configuring a Filter to Accept OSPF Packets from a Prefix | 1189

Example: Configuring a Filter to Accept DHCP Packets Based on Address | 1185

Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags

IN THIS SECTION

- Requirements | 1091
- Overview | 1091
- Configuration | 1091
- Verification | 1094

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a filter that accepts packets with specific IPv6 TCP flags.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1092
- Configure the Stateless Firewall Filter | 1092
- Apply the Firewall Filter to the Loopback Interface | 1093
- Confirm and Commit Your Candidate Configuration | 1093

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet6 filter tcp_filter term 1 from next-header tcp
set firewall family inet6 filter tcp_filter term 1 from tcp-flags syn
set firewall family inet6 filter tcp_filter term 1 then count tcp_syn_pkt
set firewall family inet6 filter tcp_filter term 1 then log
set firewall family inet6 filter tcp_filter term 1 then accept
set interfaces lo0 unit 0 family inet6 filter input tcp_filter
set interfaces lo0 unit 0 family inet6 address ::10.34.1.0/120
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the firewall filter

1. Create the IPv6 stateless firewall filter **tcp_filter**.

```
[edit]
user@host# edit firewall family inet6 filter tcp_filter
```

2. Specify that a packet matches if it is the initial packet in a TCP session and the next header after the IPv6 header is type TCP.

```
[edit firewall family inet6 filter tcp_filter]
user@host# set term 1 from next-header tcp
user@host# set term 1 from tcp-flags syn
```

3. Specify that matched packets are counted, logged to the buffer on the Packet Forwarding Engine, and accepted.

```
[edit firewall family inet6 filter tcp_filter]
user@host# set term 1 then count tcp_syn_pkt
user@host# set term 1 then log
user@host# set term 1 then accept
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

- ```
  [edit]
  user@host# set interfaces lo0 unit 0 family inet6 filter input tcp_filter
  user@host# set interfaces lo0 unit 0 family inet6 address ::10.34.1.0/120
  ```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show firewall
   family inet6 {
       filter tcp_filter {
           term 1 {
               from {
                   next-header tcp;
                   tcp-flags syn;
               }
               then {
                   count tcp_syn_pkt;
                   log;
                   accept;
               }
           }
       }
   }
   ```
   ```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet6 {
            filter {
                input tcp_filter;
            }
            address ::10.34.1.0/120;
        }
    }
}
```

3. When you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the **show firewall** operational mode command.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | **778**
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods | **1103**
- Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers | **1095**

Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers

IN THIS SECTION

- Requirements | 1095
- Overview | 1095
- Configuration | 1096
- Verification | 1101

This example shows how to configure a standard stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except from specified BGP peers.

Requirements

No special configuration beyond device initialization is required before you configure this example.

Overview

IN THIS SECTION

- Topology | 1095

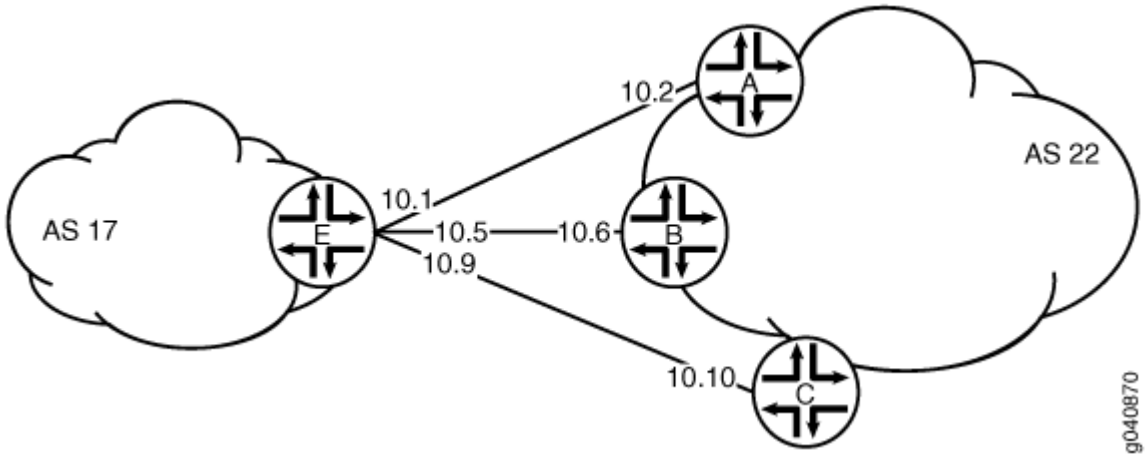
In this example, you create a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except the specified BGP peers.

The stateless firewall filter **filter_bgp179** matches all packets from the directly connected interfaces on Device A and Device B to the destination port number 179.

Topology

Figure 52 on page 1096 shows the topology used in this example. Device C attempts to make a TCP connection to Device E. Device E blocks the connection attempt. This example shows the configuration on Device E.

Figure 52: Typical Network with BGP Peer Sessions



Configuration

IN THIS SECTION

- CLI Quick Configuration | **1096**
- Configuring Device E | **1097**

CLI Quick Configuration

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

Device C

```
set interfaces ge-1/2/0 unit 10 description to-E
set interfaces ge-1/2/0 unit 10 family inet address 10.10.10.10/30
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 17
set protocols bgp group external-peers neighbor 10.10.10.9
set routing-options autonomous-system 22
```

Device E

```
set interfaces ge-1/2/0 unit 0 description to-A
set interfaces ge-1/2/0 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/2/1 unit 5 description to-B
set interfaces ge-1/2/1 unit 5 family inet address 10.10.10.5/30
set interfaces ge-1/0/0 unit 9 description to-C
set interfaces ge-1/0/0 unit 9 family inet address 10.10.10.9/30
set interfaces lo0 unit 2 family inet filter input filter_bgp179
set interfaces lo0 unit 2 family inet address 192.168.0.1/32
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 22
set protocols bgp group external-peers neighbor 10.10.10.2
set protocols bgp group external-peers neighbor 10.10.10.6
set protocols bgp group external-peers neighbor 10.10.10.10
set routing-options autonomous-system 17
set firewall family inet filter filter_bgp179 term 1 from source-address 10.10.10.2/32
set firewall family inet filter filter_bgp179 term 1 from source-address 10.10.10.6/32
set firewall family inet filter filter_bgp179 term 1 from destination-port bgp
set firewall family inet filter filter_bgp179 term 1 then accept
set firewall family inet filter filter_bgp179 term 2 then reject
```

Configuring Device E

Step-by-Step Procedure

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

To configure Device E with a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requestors except specified BGP peers:

1. Configure the interfaces.

```
user@E# set interfaces ge-1/2/0 unit 0 description to-A
user@E# set interfaces ge-1/2/0 unit 0 family inet address 10.10.10.1/30
user@E# set interfaces ge-1/2/1 unit 5 description to-B
user@E# set interfaces ge-1/2/1 unit 5 family inet address 10.10.10.5/30
user@E# set interfaces ge-1/0/0 unit 9 description to-C
user@E# set interfaces ge-1/0/0 unit 9 family inet address 10.10.10.9/30
```

2. Configure BGP.

```
[edit protocols bgp group external-peers]
user@E# set type external
user@E# set peer-as 22
user@E# set neighbor 10.10.10.2
user@E# set neighbor 10.10.10.6
user@E# set neighbor 10.10.10.10
```

3. Configure the autonomous system number.

```
[edit routing-options]
user@E# set autonomous-system 17
```

4. Define the filter term that accepts TCP connection attempts to port 179 from the specified BGP peers.

```
[edit firewall family inet filter filter_bgp179]
user@E# set term 1 from source-address 10.10.10.2/32
user@E# set term 1 from source-address 10.10.10.6/32
user@E# set term 1 from destination-port bgp
user@E# set term 1 then accept
```

5. Define the other filter term to reject packets from other sources.

```
[edit firewall family inet filter filter_bgp179]
user@E# set term 2 then reject
```

6. Apply the firewall filter to the loopback interface.

```
[edit interfaces lo0 unit 2 family inet]
user@E# set filter input filter_bgp179
user@E# set address 192.168.0.1/32
```

Results

From configuration mode, confirm your configuration by entering the **show firewall**, **show interfaces**, **show protocols**, and **show routing-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@E# show firewall
family inet {
    filter filter_bgp179 {
        term 1 {
            from {
                source-address {
                    10.10.10.2/32;
                    10.10.10.6/32;
                }
                destination-port bgp;
            }
            then accept;
        }
        term 2 {
            then {
                reject;
            }
        }
    }
}
```

```
user@E# show interfaces
lo0 {
    unit 2 {
        family inet {
            filter {
                input filter_bgp179;
            }
            address 192.168.0.1/32;
        }
    }
}
ge-1/2/0 {
    unit 0 {
        description to-A;
```

```
        family inet {
            address 10.10.10.1/30;
        }
    }
}
ge-1/2/1 {
    unit 5 {
        description to-B;
        family inet {
            address 10.10.10.5/30;
        }
    }
}
ge-1/0/0 {
    unit 9 {
        description to-C;
        family inet {
            address 10.10.10.9/30;
        }
    }
}
```

```
user@E# show protocols
bgp {
    group external-peers {
        type external;
        peer-as 22;
        neighbor 10.10.10.2;
        neighbor 10.10.10.6;
        neighbor 10.10.10.10;
    }
}
```

```
user@E# show routing-options
autonomous-system 17;
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying That the Filter Is Configured | 1101
- Verifying the TCP Connections | 1101
- Monitoring Traffic on the Interfaces | 1102

Confirm that the configuration is working properly.

Verifying That the Filter Is Configured

Purpose

Make sure that the filter is listed in output of the `show firewall filter` command.

Action

```
user@E> show firewall filter filter_bgp179
Filter: filter_bgp179
```

Verifying the TCP Connections

Purpose

Verify the TCP connections.

Action

From operational mode, run the `show system connections extensive` command on Device C and Device E.

The output on Device C shows the attempt to establish a TCP connection. The output on Device E shows that connections are established with Device A and Device B only.

```
user@C> show system connections extensive | match 10.10.10

tcp4       0      0  10.10.10.9.51872     10.10.10.10.179       SYN_SENT
```

```
user@E> show system connections extensive | match 10.10.10

tcp4       0      0  10.10.10.5.179        10.10.10.6.62096     ESTABLISHED
tcp4       0      0  10.10.10.6.62096      10.10.10.5.179       ESTABLISHED
tcp4       0      0  10.10.10.1.179        10.10.10.2.61506     ESTABLISHED
tcp4       0      0  10.10.10.2.61506      10.10.10.1.179       ESTABLISHED
```

Monitoring Traffic on the Interfaces

Purpose

Use the **monitor traffic** command to compare the traffic on an interface that establishes a TCP connection with the traffic on an interface that does not establish a TCP connection.

Action

From operational mode, run the **monitor traffic** command on the Device E interface to Device B and on the Device E interface to Device C. The following sample output verifies that in the first example, acknowledgment (**ack**) messages are received. In the second example, **ack** messages are not received.

```
user@E> monitor traffic size 1500 interface ge-1/2/1.5
19:02:49.700912 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: P 3330573561:3330573580(19) ack
915601686 win 16384 <nop,nop,timestamp 1869518816 1869504850>: BGP, length: 19
19:02:49.801244  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: . ack 19 win 16384 <nop,nop,timestamp
1869518916 1869518816>
19:03:03.323018  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: P 1:20(19) ack 19 win 16384
<nop,nop,timestamp 1869532439 1869518816>: BGP, length: 19
19:03:03.422418 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: . ack 20 win 16384 <nop,nop,timestamp
1869532539 1869532439>
19:03:17.220162 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: P 19:38(19) ack 20 win 16384
<nop,nop,timestamp 1869546338 1869532439>: BGP, length: 19
```

```
19:03:17.320501  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: . ack 38 win 16384 <nop,nop,timestamp
1869546438 1869546338>
```

```
user@E> monitor traffic size 1500 interface ge-1/0/0.9

18:54:20.175471 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,nop,wscale 0,nop,nop,timestamp 1869009240 0,sackOK,eol>
18:54:23.174422 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,nop,wscale 0,nop,nop,timestamp 1869012240 0,sackOK,eol>
18:54:26.374118 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,nop,wscale 0,nop,nop,timestamp 1869015440 0,sackOK,eol>
18:54:29.573799 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,sackOK,eol>
18:54:32.773493 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,sackOK,eol>
18:54:35.973185 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0) win 16384
<mss 1460,sackOK,eol>
```

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | **778**
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods | **1103**
- Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags | **1091**

Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods

IN THIS SECTION

- Requirements | **1104**
- Overview | **1104**
- Configuration | **1105**
- Verification | **1112**

This example shows how to create a stateless firewall filter that protects against TCP and ICMP denial-of-service attacks.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

In this example we create a stateless firewall filter called `protect-RE` to police TCP and ICMP packets. It uses the policers described here:

- `tcp-connection-policer`—This policer limits TCP traffic to 1,000,000 bits per second (bps) with a maximum burst size of 15,000 bytes. Traffic exceeding either limit is discarded.
- `icmp-policer`—This policer limits ICMP traffic to 1,000,000 bps with a maximum burst size of 15,000 bytes. Traffic exceeding either limit is discarded.

When specifying limits, the bandwidth limit can be from 32,000 bps to 32,000,000,000 bps and the burst-size limit can be from 1,500 bytes through 100,000,000 bytes. Use the following abbreviations when specifying limits: k (1,000), m (1,000,000), and g (1,000,000,000).

Each policer is incorporated into the action of a filter term. This example includes the following terms:

- `tcp-connection-term`—Polices certain TCP packets with a source address of 192.168.0.0/24 or 10.0.0.0/24. These addresses are defined in the `trusted-addresses` prefix list.

  Filtered packets include `tcp-established` packets The `tcp-established` match condition is an alias for the bit-field match condition `tcp-flags "(ack | rst)"`, which indicates an established TCP session, but not the first packet of a TCP connection.

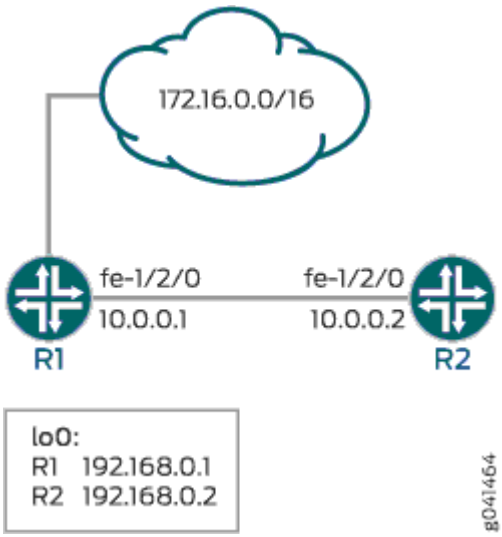
- `icmp-term`—Polices ICMP packets. All ICMP packets are counted in the `icmp-counter` counter.

NOTE: You can move terms within the firewall filter by using the `insert` command. See *insert* in the Junos OS CLI User Guide.

You can apply a stateless firewall to the input or output sides, or both, of an interface. To filter packets transiting the device, apply the firewall filter to any non-Routing Engine interface. To filter packets originating from, or destined for, the Routing Engine, apply the firewall filter to the loopback (lo0) interface.

Figure 53 on page 1105 shows the sample network.

Figure 53: Firewall Filter to Protect Against TCP and ICMP Floods



Because this firewall filter limits Routing Engine traffic to TCP packets, routing protocols that use other transport protocols for Layer 4 cannot successfully establish sessions when this filter is active. To demonstrate, this example sets up OSPF between Device R1 and Device R2.

"CLI Quick Configuration" on page 1105 shows the configuration for all of the devices in Figure 53 on page 1105.

The section "No Link Title" on page 1107 describes the steps on Device R2.

Configuration

IN THIS SECTION

- Procedure | **1105**

Procedure

CLI Quick Configuration

To quickly configure the stateless firewall filter, copy the following commands to a text file, remove any line breaks, and then paste the commands into the CLI.

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32 primary
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet filter input protect-RE
set interfaces lo0 unit 0 family inet address 192.168.0.2/32 primary
set interfaces lo0 unit 0 family inet address 172.16.0.2/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set policy-options prefix-list trusted-addresses 10.0.0.0/24
set policy-options prefix-list trusted-addresses 192.168.0.0/24
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 200
set firewall family inet filter protect-RE term tcp-connection-term from source-prefix-list
trusted-addresses
set firewall family inet filter protect-RE term tcp-connection-term from protocol tcp
set firewall family inet filter protect-RE term tcp-connection-term from tcp-established
set firewall family inet filter protect-RE term tcp-connection-term then policer tcp-connection-
policer
set firewall family inet filter protect-RE term tcp-connection-term then accept
set firewall family inet filter protect-RE term icmp-term from source-prefix-list trusted-
```

```
addresses
set firewall family inet filter protect-RE term icmp-term from protocol icmp
set firewall family inet filter protect-RE term icmp-term then policer icmp-policer
set firewall family inet filter protect-RE term icmp-term then count icmp-counter
set firewall family inet filter protect-RE term icmp-term then accept
set firewall policer tcp-connection-policer filter-specific
set firewall policer tcp-connection-policer if-exceeding bandwidth-limit 1m
set firewall policer tcp-connection-policer if-exceeding burst-size-limit 15k
set firewall policer tcp-connection-policer then discard
set firewall policer icmp-policer filter-specific
set firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set firewall policer icmp-policer if-exceeding burst-size-limit 15k
set firewall policer icmp-policer then discard
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure stateless firewall filter to discard :

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 0 family inet ]
user@R2# set address 10.0.0.2/30
[edit interfaces lo0 unit 0 family inet]
user@R2# set address 192.168.0.2/32 primary
user@R2# set address 172.16.0.2/32
```

2. Configure the BGP peering session.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set export send-direct
user@R2# set neighbor 10.0.0.1 peer-as 100
```

3. Configure the autonomous system (AS) number and router ID.

```
[edit routing-options]
user@R2# set autonomous-system 200
user@R2# set router-id 192.168.0.2
```

4. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R2# set interface lo0.0 passive
user@R2# set interface fe-1/2/0.0
```

5. Define the list of trusted addresses.

```
[edit policy-options prefix-list trusted-addresses]
user@R2# set 10.0.0.0/24
user@R2# set 192.168.0.0/24
```

6. Configure a policy to advertise direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

7. Configure the TCP policer.

```
[edit firewall policer tcp-connection-policer]
user@R2# set filter-specific
user@R2# set if-exceeding bandwidth-limit 1m
user@R2# set if-exceeding burst-size-limit 15k
user@R2# set then discard
```

8. Create the ICMP policer.

```
[edit firewall policer icmp-policer]
user@R2# set filter-specific
user@R2# set if-exceeding bandwidth-limit 1m
```

```
user@R2# set if-exceeding burst-size-limit 15k
user@R2# set then discard
```

9. Configure the TCP filter rules.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# set from source-prefix-list trusted-addresses
user@R2# set from protocol tcp
user@R2# set from tcp-established
user@R2# set then policer tcp-connection-policer
user@R2# set then accept
```

10. Configure the ICMP filter rules.

```
[edit firewall family inet filter protect-RE term icmp-term]
user@R2# set from source-prefix-list trusted-addresses
user@R2# set from protocol icmp
user@R2# set then policer icmp-policer
user@R2# set then count icmp-counter
user@R2# set then accept
```

11. Apply the filter to the loopback interface.

```
[edit interfaces lo0 unit 0]
user@R2# set family inet filter input protect-RE
```

Results

Confirm your configuration by entering the `show interfaces`, `show protocols`, `show policy-options`, `show routing-options`, and `show firewall` commands from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
```

```
    }
}
lo0 {
    unit 0 {
        family inet {
            filter {
                input protect-RE;
            }
            address 192.168.0.2/32 {
                primary;
            }
            address 172.16.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group ext {
        type external;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-1/2/0.0;
    }
}
```

```
user@R2# show policy-options
prefix-list trusted-addresses {
    10.0.0.0/24;
    192.168.0.0/24;
}
```

```
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
}
```

```
user@R2# show routing-options
router-id 192.168.0.2;
autonomous-system 200;
```

```
user@R2# show firewall
family inet {
    filter protect-RE {
        term tcp-connection-term {
            from {
                source-prefix-list {
                    trusted-addresses;
                }
                protocol tcp;
                tcp-established;
            }
            then {
                policer tcp-connection-policer;
                accept;
            }
        }
        term icmp-term {
            from {
                source-prefix-list {
                    trusted-addresses;
                }
                protocol icmp;
            }
            then {
                policer icmp-policer;
                count icmp-counter;
                accept;
            }
        }
```

```
    }
}
policer tcp-connection-policer {
    filter-specific;
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
}
policer icmp-policer {
    filter-specific;
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then discard;
}
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Stateless Firewall Filter That Are in Effect | **1113**
- Using telnet to Verify the tcp-established Condition in the TCP Firewall Filter | **1113**
- Using telnet to Verify the Trusted Prefixes Condition in the TCP Firewall Filter | **1115**
- Using OSPF to Verify the TCP Firewall Filter | **1116**
- Verifying the ICMP Firewall Filter | **1117**

Confirm that the configuration is working properly.

NOTE: To verify the TCP policer, you can use a packet generation tool. This task is not shown here.

Displaying Stateless Firewall Filter That Are in Effect

Purpose

Verify the configuration of the firewall filter.

Action

From operational mode, enter the `show firewall` command.

```
user@R2> show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
icmp-counter                                            0                    0
Policers:
Name                                                Bytes              Packets
icmp-policer                                                            0
tcp-connection-policer                                                  0
```

Meaning

The output shows the filter, the counter, and the policers that are in effect on Device R2.

Using telnet to Verify the tcp-established Condition in the TCP Firewall Filter

Purpose

Make sure that telnet traffic works as expected.

Action

Verify that the device can establish only TCP sessions with hosts that meet the `from tcp-established` condition.

1. From Device R2, make sure that the BGP session with Device R1 is established.

```
user@R2> show bgp summary | match down
Groups: 1 Peers: 1 Down peers: 0
```

2. From Device R2, telnet to Device R1.

```
user@R2> telnet 192.168.0.1
Trying 192.168.0.1...
Connected to R1.example.net.
Escape character is '^]'.

R1 (ttyp4)

login:
```

3. From Device R1, telnet to Device R2.

```
user@R1> telnet 192.168.0.2
Trying 192.168.0.2...
telnet: connect to address 192.168.0.2: Operation timed out
telnet: Unable to connect to remote host
```

4. On Device R2, deactivate the `from tcp-established` match condition.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# deactivate from tcp-established
user@R2# commit
```

5. From Device R1, try again to telnet to Device R2.

```
user@R1> telnet 192.168.0.1
Trying 192.168.0.2...
Connected to R2.example.net.
Escape character is '^]'.

R2 (ttyp4)

login:
```

Meaning

Verify the following information:

- As expected , the BGP session is established. The `from tcp-established` match condition is not expected to block BGP session establishment.
- From Device R2, you can telnet to Device R1. Device R1 has no firewall filter configured, so this is the expected behavior.
- From Device R1, you cannot telnet to Device R2. Telnet uses TCP as the transport protocol, so this result might be surprising. The cause for the lack of telnet connectivity is the `from tcp-established` match condition. This match condition limits the type of TCP traffic that is accepted of Device R2. After this match condition is deactivated, the telnet session is successful.

Using telnet to Verify the Trusted Prefixes Condition in the TCP Firewall Filter

Purpose

Make sure that telnet traffic works as expected.

Action

Verify that the device can establish only telnet sessions with a host at an IP address that matches one of the trusted source addresses. For example, log in to the device with the `telnet` command from another host with one of the trusted address prefixes. Also, verify that telnet sessions with untrusted source addresses are blocked.

1. From Device R1, telnet to Device R2 from an untrusted source address.

```
user@R1> telnet 172.16.0.2 source 172.16.0.1
Trying 172.16.0.2...
^C
```

2. From Device R2, add 172.16/16 to the list of trusted prefixes.

```
[edit policy-options prefix-list trusted-addresses]
user@R2# set 172.16.0.0/16
user@R2# commit
```

3. From Device R1, try again to telnet to Device R2.

```
user@R1> telnet 172.16.0.2 source 172.16.0.1
Trying 172.16.0.2...
Connected to R2.example.net.
```

```
Escape character is '^]'.

R2 (ttyp4)

login:
```

Meaning

Verify the following information:

- From Device R1, you cannot telnet to Device R2 with an unstrusted source address. After the 172.16/16 prefix is added to the list of trusted prefixes, the telnet request from source address 172.16.0.1 is accepted.
- OSPF session establishment is blocked. OSPF does not use TCP as its transport protocol. After the `from protocol tcp` match condition is deactivated, OSPF session establishment is not blocked.

Using OSPF to Verify the TCP Firewall Filter

Purpose

Make sure that OSPF traffic works as expected.

Action

Verify that the device cannot establish OSPF connectivity.

1. From Device R1, check the OSPF sessions.

```
user@R1> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.2         fe-1/2/0.0             Init      192.168.0.2      128    34
```

2. From Device R2, check the OSPF sessions.

```
user@R2> show ospf neighbor
```

3. From Device R2, remove the `from protocol tcp` match condition.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# deactivate from protocol
user@R2# commit
```

4. From Device R1, recheck the OSPF sessions.

```
user@R1> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.2         fe-1/2/0.0             Full      192.168.0.2      128    36
```

5. From Device R2, recheck the OSPF sessions.

```
user@R2> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.1         fe-1/2/0.0             Full      192.168.0.1      128    39
```

Meaning

Verify the following information:

- OSPF session establishment is blocked. OSPF does not use TCP as its transport protocol. After the `from protocol tcp` match condition is deactivated, OSPF session establishment is successful.

Verifying the ICMP Firewall Filter

Purpose

Verify that ICMP packets are being policed and counted. Also make sure that ping requests are discarded when the requests originate from an untrusted source address.

Action

1. Undo the configuration changes made in previous verification steps.

Reactivate the TCP firewall settings, and delete the 172.16/16 trusted source address.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# activate from protocol
user@R2# activate from tcp-established

[edit policy-options prefix-list trusted-addresses]
user@R2# delete 172.16.0.0/16

user@R2# commit
```

2. From Device R1, ping the loopback interface on Device R2.

```
user@R1> ping 192.168.0.2 rapid count 600 size 2000
PING 192.168.0.2 (192.168.0.2): 2000 data bytes
!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!
!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!
!!!!!!!.!!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!
!!!!.!!!!!!!!!.!!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!
!!!.!!!!!!!!.!!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!
!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!.
!!!!!!!!!.!!!!!!!!.!!!!!!!!.!!!!!!!!
--- 192.168.0.2 ping statistics ---
600 packets transmitted, 536 packets received, 10% packet loss
pinground-trip min/avg/max/stddev = 2.976/3.405/42.380/2.293 ms
```

3. From Device R2, check the firewall statistics.

```
user@R2> show firewall

Filter: protect-RE
Counters:
Name                                                Bytes              Packets
icmp-counter                                      1180804                 1135
Policers:
Name                                                Bytes              Packets
icmp-policer                                                           66
tcp-connection-policer                                                  0
```

4. From an untrusted source address on Device R1, send a ping request to Device R2's loopback interface.

```
user@R1> ping 172.16.0.2 source 172.16.0.1

PING 172.16.0.2 (172.16.0.2): 56 data bytes
^C
--- 172.16.0.2 ping statistics ---
14 packets transmitted, 0 packets received, 100% packet loss
```

Meaning

Verify the following information:

- The ping output shows that 10% packet loss is occurring.
- The ICMP packet counter is incrementing, and the icmp-policer is incrementing.
- Device R2 does not send ICMP responses to the `ping 172.16.0.2 source 172.16.0.1` command.

RELATED DOCUMENTATION

Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | 1065

Two-Color Policer Configuration Overview | 2034

Example: Protecting the Routing Engine with a Packets-Per-Second Rate Limiting Filter

IN THIS SECTION

- Requirements | 1120
- Overview | 1120
- Configuration | 1120
- Verification | 1123

This example shows how to configure a packets-per-second based rate-limiting filter to improve security. It will be applied to the loopback interface in order to help protect the Routing Engine from denial of service attacks.

> **BEST PRACTICE**: This type of filter and policer combination is only one element in a multilayered approach that can be used to help protect the Routing Engine. Other layers of protection are needed in order to fully protect the Routing Engine. See Day One: Securing the Routing Engine on M, MX, and T Series for more information.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to set packets-per-second (pps) rate limits for any traffic destined for the Routing Engine through the loopback interface (lo0.0).

To activate a policer from within a stateless firewall filter configuration:

1. Create a template for the policer by including the `policer` *policer-name* statement at the `[edit firewall]` hierarchy.
2. Reference the policer in a filter term that specifies the policer in the `policer` *policer-name* nonterminating action.

You can also apply a policer by including the `policer (input | output)` *policer-name* statement in a logical interface configuration.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1121**
- Configuring the Policer and the Stateless Firewall Filter | **1121**
- Applying the Stateless Firewall Filter to the Loopback Logical Interface | **1122**
- Results | **1122**

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

CLI Quick Configuration

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

```
set firewall policer police_pps if-exceeding-pps pps-limit 1k
set firewall policer police_pps if-exceeding-pps packet-burst 150
set firewall policer police_pps then discard
set firewall family inet filter my_pps_filter term term1 then policer police_pps
set interfaces lo0 unit 0 family inet filter input my_pps_filter
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configuring the Policer and the Stateless Firewall Filter

Step-by-Step Procedure

To configure the policer *police_pps* and stateless firewall filter *my_pps_filter*:

1. Configure the policer template `police_pps`.

```
[edit firewall]
user@host# set policer police_pps if-exceeding-pps pps-limit 1k
user@host# set policer police_pps if-exceeding-pps packet-burst 150
user@host# set policer police_pps then discard
```

2. Create the stateless firewall filter `my_pps_filter`.

```
[edit]
user@host# edit firewall family inet filter my_pps_filter
```

3. Configure a filter term that uses policer `police_pps` to rate limit traffic by protocol family.

```
[edit firewall family inet filter my_pps_filter]
user@host# set term term1 then policer police_pps
```

Applying the Stateless Firewall Filter to the Loopback Logical Interface

Step-by-Step Procedure

To apply the filter `my_pps_filter` to the loopback interface:

1. Configure the logical loopback interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces lo0 unit 0
```

2. Apply the stateless firewall filter to the loopback interface.

```
[edit interfaces lo0 unit 0]
user@host# set family inet filter input my_pps_filter
```

3. Configure the interface address for the loopback interface.

```
[edit interfaces lo0 unit 0]
user@host# set family inet address 127.0.0.1/32
```

Results

Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet{
    filter my_pps_filter {
        term term1 {
            then policer police_pps;
```

```
        }
    }
}
policer police_pps {
    if-exceeding-pps {
        pps-limit 1k;
        packet-burst 150;
    }
    then discard;
}
```

Confirm the configuration of the interface by entering the `show interfaces lo0` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show interfaces lo0
unit 0 {
    family inet {
        filter {
            input my_pps_filter;
        }
        address 127.0.0.1/32;
    }
}
```

If you are done configuring the device, enter commit from configuration mode.

```
user@host# commit
```

Verification

IN THIS SECTION

- Verifying the Operation of the Filter | **1124**

Verifying the Operation of the Filter

Purpose

To confirm that the configuration is working properly, enter the `show firewall filter my_pps_filter` operational mode command.

> **NOTE**: The following output results from running a rapid ping from another host to the router under test. In order to show results in the output, a `pps-limit` setting of **50** and a `packet-burst` setting of **10** were used during the ping test.

Action

```
user@host> show firewall filter my_pps_filter
Filter: my_pps_filter
Policers:
Name                                                Bytes              Packets
police_pps-term1                                     8704                   17
```

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | **778**
- Packets-Per-Second (pps)-Based Policer Overview | **1934**
- *if-exceeding-pps (Policer)*

Example: Configuring a Filter to Exclude DHCPv6 and ICMPv6 Control Traffic for LAC Subscriber

IN THIS SECTION

- Requirements | **1125**
- Overview | **1125**

- Configuration | 1125

This example shows how to configure a standard stateless firewall filter that excludes DHCPv6 and ICMPv6 control packets from being considered for idle-timeout detection for tunneled subscribers at the LAC.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

Subscriber access on a LAC can be limited by configuring an idle timeout period that specifies the maximum period of time a subscriber can remain idle after the subscriber session is established. The LAC monitors the subscriber's upstream and downstream data traffic to determine whether the subscriber is inactive. Based on the session accounting statistics. the subscriber is not considered idle as long as data traffic is detected in either direction. When no traffic is detected for the duration of the idle time out, the subscriber is logged out gracefully similarly to a RADIUS-initiated disconnect or a CLI-initiated logout.

However, after a tunnel is established for L2TP subscribers, all packets through the tunnel at the LAC are treated as data packets. Consequently, the accounting statistics for the session are inaccurate and the subscriber is not considered to be idle as long as DHCPv6 and ICMPv6 control packets are being sent.

Starting in Junos OS Release 17.2R1, you can define a firewall filter for the `inet6` family with terms to match on these control packets. Include the use the `exclude-accounting` terminating action in the filter terms to drop these control packets.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1126
- Configure the Filter | 1126
- Results | 1129

CLI Quick Configuration

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

```
set access profile v6-exclude-idle session-options client-idle-timeout 10
set access profile v6-exclude-idle session-options client-idle-timeout-ingress-only

edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER
set interface-specific
set term EXCLUDE-ACCT-DHCP-INET6 from next-header udp
set term EXCLUDE-ACCT-DHCP-INET6 from source-port 546
set term EXCLUDE-ACCT-DHCP-INET6 from source-port 547
set term EXCLUDE-ACCT-DHCP-INET6 from destination-port 546
set term EXCLUDE-ACCT-DHCP-INET6 from destination-port 547
set term EXCLUDE-ACCT-DHCP-INET6 then count exclude-acct-dhcpv6
set term EXCLUDE-ACCT-DHCP-INET6 then exclude-accounting

set term EXCLUDE-ACCT-ICMP6 from next-header icmp6
set term EXCLUDE-ACCT-ICMP6 from icmp-type router-solicit
set term EXCLUDE-ACCT-ICMP6 from icmp-type neighbor-solicit
set term EXCLUDE-ACCT-ICMP6 from icmp-type neighbor-advertisement
set term EXCLUDE-ACCT-ICMP6 then count exclude-acct-icmpv6
set term EXCLUDE-ACCT-ICMP6 then exclude-accounting

set term default then accept

top
edit dynamic-profiles pppoe-dynamic-profile interfaces pp0 unit "$junos-interface-unit"
set family inet6 filter input EXCLUDE-ACCT-INET6-FILTER
set family inet6 filter output EXCLUDE-ACCT-INET6-FILTER
set actual-transit-statistics
```

Configure the Filter

Step-by-Step Procedure

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

To configure the filter:

1. Set the idle timeout for subscriber sessions..

```
[edit access profile v6-exclude-idle]
user@host# set session-options client-idle-timeout 10
```

2. Specify the idle timeout applies only to ingress traffic.

```
[edit access profile v6-exclude-idle]
user@host# set session-options client-idle-timeout-ingress-only
```

3. Define the firewall filter term that excludes the DHCPv6 control packets from accounting statistics.

   a. Specify a match on packets with the first Next Header field set to UDP (17).

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 from next-header udp
```

   b. Specify a match on packets with a source port of 546 or 547 (DHCPv6).

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 from source-port 546
user@host# set term EXCLUDE-ACCT-DHCP-INET6 from source-port 547
```

   c. Specify a match on packets with a DHCP destination port of 546 or 547 (DHCPv6).

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 from destination-port 546
user@host# set term EXCLUDE-ACCT-DHCP-INET6 from destination-port 547
```

   d. Count the matched DHCPv6 packets.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 then count exclude-acct-dhcpv6
```

e. Exclude the matched DHCPv6 packets from accounting statistics.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 then exclude-accounting
```

4. Define the firewall filter term that excludes the ICMPv6 control packets from accounting statistics.

a. Specify a match on packets with the first Next Header field set to ICMPv6 (58).

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-ICMP6 from next-header icmp6
```

b. Specify a match on packets with an ICMPv6 message type.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-ICMP6 from icmp-type router-solicit
user@host# set term EXCLUDE-ACCT-ICMP6 from icmp-type neighbor-solicit
user@host# set term EXCLUDE-ACCT-ICMP6 from icmp-type neighbor-advertisement
```

c. Count the matched ICMPv6 packets.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-ICMP6 then count exclude-acct-icmpv6
```

d. Exclude the matched ICMPv6 packets from accounting statistics.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term EXCLUDE-ACCT-DHCP-INET6 then exclude-accounting
```

5. Define the default filter term to accept all other packets.

```
[edit firewall family inet6 filter EXCLUDE-ACCT-INET6-FILTER]
user@host# set term default then accept
```

6. Configure the dynamic profile to apply the filter to input and output interfaces for the `inet6` family.

```
[edit dynamic-profiles pppoe-dynamic-profile interfaces pp0 unit "$junos-interface-unit"]
user@host# set family inet6 filter input EXCLUDE-ACCT-INET6-FILTER
user@host# set family inet6 filter output EXCLUDE-ACCT-INET6-FILTER
```

7. Enable subscriber management accurate accounting.

```
[edit dynamic-profiles pppoe-dynamic-profile interfaces pp0 unit "$junos-interface-unit"]
user@host# set actual-transit-statistics
```

Results

From configuration mode, confirm your configuration by entering the `show access`, `show firewall`, and `show dynamic-profiles` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show access
profile v6-exclude-idle {
    session-options {
        client-idle-timeout 10;
        client-idle-timeout-ingress-only;
    }
}
```

```
user@host# show firewall
family inet6 {
    filter EXCLUDE-ACCT-INET6-FILTER {
        interface-specific;
        term EXCLUDE-ACCT-DHCP-INET6 {
            from {
                next-header udp;
                 source-port [ 546 547 ];
                destination-port [ 546 547 ];
            }
            then {
                count exclude-acct-dhcpv6;
                exclude-accounting
```

```
            }
        }
        term EXCLUDE-ACCT-ICMP6 {
            from {
                next-header icmp6;
                 icmp-type [ router-solicit neighbor-solicit neighbor-advertisement ]
            }
            then {
                count exclude-acct-icmpv6;
                exclude-accounting;
            }
        }
        term default {
            then accept;
        }
    }
}
```

```
user@host# show dynamic-profiles
pppoe-dynamic-profile {
    interfaces {
        pp0 {
            unit "$junos-interface-unit" {
                actual-transit-statistics;
                family inet6 {
                    filter {
                        input EXCLUDE-ACCT-INET6-FILTER;
                        output EXCLUDE-ACCT-INET6-FILTER;
                    }
                }
            }
        }
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

RELATED DOCUMENTATION

Classic Filters Overview

Dynamically Attaching Statically Created Filters for a Specific Interface Family Type

Understanding How to Use Standard Firewall Filters

Firewall Filter Terminating Actions

Firewall Filter Match Conditions for IPv6 Traffic

Port Number Requirements for DHCP Firewall Filters

When you configure a *firewall filter* to perform some action on DHCP packets at the Routing Engine, such as protecting the Routing Engine by allowing only proper DHCP packets, you must specify both port 67 (bootps) and port 68 (bootpc) for both the source and destination. The firewall filter acts at both the line cards and the Routing Engine.

This requirement applies to both DHCP local server and DHCP relay, but it applies only when DHCP is provided by the jdhcpd process. MX Series routers use jdhcpd. For DHCP relay, that means the configuration is required only at the `[edit forwarding-options dhcp-relay]` hierarchy level and not at the `[edit forwarding-options helpers bootp]` hierarchy level.

DHCP packets received on the line cards are encapsulated by jdhcpd with a new UDP header where their source and destination addresses are set to port 68 before being forwarded to the Routing Engine.

For DHCP relay and DHCP proxy, packets sent to the DHCP server from the router have both the source and destination UDP ports set to 67. The DHCP server responds using the same ports. However, when the line card receives these DHCP response packets, it changes both port numbers from 67 to 68 before passing the packets to the Routing Engine. Consequently the filter needs to accept port 67 for packets relayed from the client to the server, and port 68 for packets relayed from the server to the client.

Failure to include both port 67 and port 68 as described here results in most DHCP packets not being accepted.

For complete information about configuring firewall filters in general, see Junos OS Routing Policies, Firewall Filters and Traffic Policers User Guide for Routing Devices.

RELATED DOCUMENTATION

Example: Configuring a DHCP Firewall Filter to Protect the Routing Engine | **1132**

Understanding Differences Between Legacy DHCP and Extended DHCP

Extended DHCP Relay Agent Overview

Understanding Dynamic Firewall Filters

Example: Configuring a DHCP Firewall Filter to Protect the Routing Engine

IN THIS SECTION

- Requirements | 1132
- Overview | 1132
- Configuration | 1133
- Verification | 1136

This example shows how to configure a firewall filter to ensure that proper DHCP packets can reach the Routing Engine on MX Series routers.

Requirements

This configuration example applies only to routers where DHCP local server and DHCP relay agent services are provided by the jdhcpd process rather than the legacy dhcpd process or fud (UDP forwarding) process. MX Series routers, M120 routers, and M320 routers use jdhcpd. For DHCP relay, that means the configuration is required only at the `[edit forwarding-options dhcp-relay]` hierarchy level and not at the `[edit forwarding-options helpers bootp]` hierarchy level.

No special configuration beyond device initialization is required before you can configure this feature.

Overview

Firewall filters that perform some action on DHCP packets at the Routing Engine, such as a filter to protect the Routing Engine by allowing only proper DHCP packets, require that both port 67 (bootps) and port 68 (bootpc) are configured as both source and destination ports.

DHCP packets received on the line cards are encapsulated by jdhcpd with a new UDP header where their source and destination addresses are set to port 68 before being forwarded to the Routing Engine. For DHCP relay and DHCP proxy, packets sent to the DHCP server from the router have both the source and destination UDP ports set to 67. The DHCP server responds using the same ports. However, when the line card receives these DHCP response packets, it changes both port numbers from 67 to 68 before passing the packets to the Routing Engine. Consequently the filter needs to accept port 67 for packets relayed from the client to the server, and port 68 for packets relayed from the server to the client.

In this example, you configure two filter terms, `dhcp-client-accept` and `dhcp-server-accept`. The match conditions for `dhcp-client-accept` specify a source address and destination address for broadcast packets,

the UDP protocol used for DHCP packets, and the bootpc (68) source port. Packets that match these conditions are counted and accepted. This term does not need to specify a match condition for the boot ps (67) destination port. As configured below, this term can handle both the actual packet (port 68) passing to the Packet Forwarding Engine and the encapsulated packet (port 67 converted to 68 by jdhcpd) that reaches the DHCP daemon.

The match conditions for `dhcp-server-accept` specify the UDP protocol used for DHCP packets, and both port 67 and 68 for both source port and destination port. Packets that match these conditions are counted and accepted.

NOTE: This example does not show all possible configuration choices, nor does it show how the filter is applied in your configuration. This example applies to both static application of the filter as well as dynamic application with a dynamic profile.

Configuration

IN THIS SECTION

- Procedure | 1133

Procedure

CLI Quick Configuration

To quickly configure the sample Routing Engine DHCP filter, copy the following commands, paste them in a text file, remove any line breaks, and then copy and paste the commands into the CLI.

```
[edit]
edit firewall family inet filter RE-protect
edit term dhcp-client-accept
set from source-address 0.0.0.0/32
set from destination-address 255.255.255.255/32
set from protocol udp
set from source-port 68
set then count dhcp-client-accept
set then accept
up
edit term dhcp-server-accept
```

```
set from protocol udp
set from source-port 67
set from source-port 68
set from destination-port 67
set from destination-port 68
set then count dhcp-server-accept
set then accept
top
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode*.

To configure a DHCP firewall filter to protect the Routing Engine:

1. Create or specify a firewall filter.

```
[edit firewall]
user@host# edit family inet filter RE-protect
```

2. Create a filter term for the client.

```
[edit firewall family inet filter RE-protect]
user@host# edit term dhcp-client-accept
```

3. Specify the match conditions for DHCP packets.

```
[edit firewall family inet filter RE-protect term dhcp-client-accept]
user@host# set from source-address 0.0.0.0/32
user@host# set from destination-address 255.255.255.255/32
user@host# set from protocol udp
user@host# set from source-port 68
user@host# set from destination-port 67
```

4. Specify the action to take for matched packets.

```
[edit firewall family inet filter RE-protect term dhcp-client-accept]
user@host# set then count dhcp-client-accept
user@host# set then accept
```

5. Create a filter term for the server.

```
[edit firewall family inet filter RE-protect]
user@host# edit term dhcp-server-accept
```

6. Specify the match conditions for DHCP packets.

```
[edit firewall family inet filter RE-protect term dhcp-server-accept]
user@host# set from protocol udp
user@host# set from source-port [67 68]
user@host# set from destination-port [67 68]
```

7. Specify the action to take for matched packets.

```
[edit firewall family inet filter RE-protect term dhcp-server-accept]
user@host# set then count dhcp-client-accept
user@host# set then accept
```

Results

From configuration mode, confirm your configuration by entering the `show firewall` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show firewall
family inet {
    filter RE-protect {
        term dhcp-client-accept {
            from {
                source-address {
                    0.0.0.0/32;
```

```
                }
                destination-address {
                    255.255.255.255/32;
                }
                protocol udp;
                source-port 68;
                destination-port 67;
            }
            then {
                count dhcp-client-accept;
                accept;
            }
        }
        term dhcp-server-accept {
            from {
                protocol udp;
                source-port [ 67 68 ];
                 destination-port [ 67 68 ];
            }
            then {
                count dhcp-server-accept;
                accept;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the DHCP Filter Operation | **1137**

To confirm that the Routing Engine DHCP protection filter is properly passing DHCP packets, perform these tasks:

Verifying the DHCP Filter Operation

Purpose

Verify that both counters increment as DHCP traffic passes to the Routing Engine.

Action

From operational mode, enter the `show firewall family inet filter RE-protect` command.

```
user@host> show firewall family inet filter RE-protect
Filter: RE-protect
Counters:
Name                                                Bytes              Packets
dhcp-client-accept                                    328                    1
dhcp-server-accept                                    574                    1

user@host> show firewall family inet filter RE-protect
Filter: RE-protect
Counters:
Name                                                Bytes              Packets
dhcp-client-accept                                    660                    2
dhcp-server-accept                                   1152                    2
```

Meaning

The output lists both configured counters, dhcp-client-accept and dhcp-server-accept. By issuing the command more than once, you can see that the byte and packet fields both show that traffic is being accepted and counted.

RELATED DOCUMENTATION

Port Number Requirements for DHCP Firewall Filters | **1131**

Understanding Dynamic Firewall Filters

Understanding Dynamic Firewall Filters

Junos OS Routing Policies, Firewall Filters and Traffic Policers User Guide for Routing Devices

Understanding Differences Between Legacy DHCP and Extended DHCP

Extended DHCP Relay Agent Overview

CHAPTER 16

Applying Firewall Filters to Transit Traffic

IN THIS CHAPTER

- Example: Configuring a Filter for Use as an Ingress Queuing Filter | **1138**
- Example: Configuring a Filter to Match on IPv6 Flags | **1142**
- Example: Configuring a Filter to Match on Port and Protocol Fields | **1144**
- Example: Configuring a Filter to Count Accepted and Rejected Packets | **1149**
- Example: Configuring a Filter to Count and Discard IP Options Packets | **1154**
- Example: Configuring a Filter to Count IP Options Packets | **1158**
- Example: Configuring a Filter to Count and Sample Accepted Packets | **1165**
- Example: Configuring a Filter to Set the DSCP Bit to Zero | **1172**
- Example: Configuring a Filter to Set the DSCP Bit to Zero | **1177**
- Example: Configuring a Filter to Match on Two Unrelated Criteria | **1181**
- Example: Configuring a Filter to Accept DHCP Packets Based on Address | **1185**
- Example: Configuring a Filter to Accept OSPF Packets from a Prefix | **1189**
- Example: Configuring a Stateless Firewall Filter to Handle Fragments | **1193**
- Configuring a Firewall Filter to Prevent or Allow IPv4 Packet Fragmentation | **1201**
- Configuring a Firewall Filter to Discard Ingress IPv6 Packets with a Mobility Extension Header | **1202**
- Example: Configuring an Egress Filter Based on IPv6 Source or Destination IP Addresses | **1203**
- Example: Configuring a Rate-Limiting Filter Based on Destination Class | **1208**

Example: Configuring a Filter for Use as an Ingress Queuing Filter

IN THIS SECTION

- Requirements | **1139**
- Overview | **1139**

Configuration | 1140

This example shows how to configure a firewall filter for use as an ingress queuing filter. The ingress queuing filter assists in traffic shaping operations by enabling you to set the forwarding class and packet loss priority, or drop the packet before ingress queue selection. The firewall filter must be configured within one of the following protocol families: `bridge`, `cc`, `inet`, `inet6`, `mpls`, or `vpls` and have one or more of the following actions: `accept`, `discard`, `forwarding-class`, and `loss-priority`.

NOTE: Although the ingress queuing filter can be used with EX9200 switches and T-Series routers as well as MX-Series routers, it is used only on those MX Series routers that have MPCs. An error is generated at commit if the ingress queuing filter is applied to an interface on any other type of port concentrator.

Requirements

This example uses the following hardware and software components:

- An MX Series router with MPC

In order for ingress queuing filters to function, `ingress-and-egress` must be configured as the `traffic-manager` mode at the `[edit chassis fpc` *`slot`* `pic` *`slot`* `traffic-manager mode]` hierarchy level.

Overview

In this example, you create a firewall filter named `iqfilter1` in the `inet` protocol family that sets the loss priority and forwarding class of packets coming from the 192.168.2.0/24 network. You then apply the `iqfilter1` filter to the ge-0/0/0.0 logical interface as an ingress queuing filter.

To configure a firewall filter and apply it for use as an ingress queuing filter involves:

- Creating a firewall filter named `iqfilter1` in the `inet` protocol family with the following two actions: `forwarding-class` and `loss-priority`.
- Applying the firewall filter to the ge-0/0/0.0 interface as an ingress queuing filter.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1140
- Configuring the Firewall Filter and Applying It to an Interface as an Input Queuing Filter | 1140
- Results | 1141

CLI Quick Configuration

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

```
set firewall family inet filter iqfilter1 term t1 from address 192.168.2.0/24
set firewall family inet filter iqfilter1 term t1 then loss-priority low
set firewall family inet filter iqfilter1 term t1 then forwarding-class expedited-forwarding
set interfaces ge-0/0/0 unit 0 family inet ingress-queuing-filter iqfilter1
```

Configuring the Firewall Filter and Applying It to an Interface as an Input Queuing Filter

Step-by-Step Procedure

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

To configure the firewall filter, `iqfilter1`, and apply it to logical interface ge-0/0/0 unit 0:

1. Create a firewall filter named `iqfilter1`.

```
[edit firewall family inet]
user@router# set  filter iqfilter1 term t1 from address 192.168.2.0/24
user@router# set  filter iqfilter1 term t1 then loss-priority low
user@router# set  filter iqfilter1 term t1 then forwarding-class expedited-forwarding
```

2. Apply the firewall filter to the logical interface.

```
[edit]
user@router# set interfaces ge-0/0/0 unit 0 family inet ingress-queuing-filter iqfilter1
```

Results

From configuration mode, confirm your configuration by entering the `show firewall` and the `show interfaces ge-0/0/0.0` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@router# show firewall
 family inet {
    filter iqfilter1 {
        term t1 {
            from {
                address {
                    192.168.0.0/24;
                }
            }
            then {
                loss-priority low;
                forwarding-class expedited-forwarding;
            }
        }
    }
}
user@router# show interfaces ge-0/0/0.0
family inet {
    ingress-queuing-filter iqfilter1;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

```
user@router# commit
```

RELATED DOCUMENTATION

Multifield Classifier for Ingress Queuing on MX Series Routers with MPC | 1330

ingress-queuing-filter

Example: Configuring a Filter to Match on IPv6 Flags

IN THIS SECTION

- Requirements | 1142
- Overview | 1142
- Configuration | 1142
- Verification | 1144

This example shows how to configure a filter to match on IPv6 TCP flags.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure a filter to match on IPv6 TCP flags. You can use this example to configure IPv6 TCP flags in M Series, MX Series, and T Series routing devices.

Configuration

IN THIS SECTION

- Procedure | 1143

Procedure

Step-by-Step Procedure

To configure a filter to match on IPv6 TCP flags:

1. Include the family statement at the firewall hierarchy level, specifying `inet6` as the protocol family.

```
[edit]
user@host# edit firewall family inet6
```

2. Create the stateless firewall filter.

```
[edit  firewall family inet6]
user@host# edit filter tcpfilt
```

3. Define the first term for the filter.

```
[edit firewall family inet6 filter tcpfilt]
user@host# edit term 1
```

4. Define the source address match conditions for the term.

```
[edit firewall family inet6 filter tcpfilt term 1]
user@host# set from next-header tcp tcp-flags syn
```

5. Define the actions for the term.

```
[edit firewall family inet6 filter tcpfilt term 1]
user@host# set then count tcp_syn_pkt log accept
```

6. If you are done configuring the device, commit the configuration.

```
[edit firewall family inet6 filter tcpfilt term 1]
user@host top
```

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter tcpfilt` command.

Example: Configuring a Filter to Match on Port and Protocol Fields

IN THIS SECTION

- Requirements | **1144**
- Overview | **1144**
- Configuration | **1144**
- Verification | **1149**

This example shows how to configure a standard stateless firewall filter to match on destination port and protocol fields.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure a stateless firewall filter that accepts all IPv4 packets except for TCP and UDP packets. TCP and UDP packets are accepted if destined for the SSH port or the Telnet port. All other packets are rejected.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1145**
- Configure the Stateless Firewall Filter | **1145**

Apply the Stateless Firewall Filter to a Logical Interface | 1146
Confirm and Commit Your Candidate Configuration | 1147

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level:

```
set firewall family inet filter filter1 term term1 from protocol-except tcp
set firewall family inet filter filter1 term term1 from protocol-except udp
set firewall family inet filter filter1 term term1 then accept
set firewall family inet filter filter1 term term2 from address 192.168.0.0/16
set firewall family inet filter filter1 term term2 then reject
set firewall family inet filter filter1 term term3 from destination-port ssh
set firewall family inet filter filter1 term term3 from destination-port telnet
set firewall family inet filter filter1 term term3 then accept
set firewall family inet filter filter1 term term4 then reject
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input filter1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `filter1`:

1. Create the IPv4 stateless firewall filter.

```
[edit]
user@host# edit firewall family inet filter filter1
```

2. Configure a term to accept all traffic except for TCP and UDP packets.

```
[edit firewall family inet filter filter1]
user@host# set term term1 from protocol-except tcp
user@host# set term term1 from protocol-except udp
user@host# set term term1 then accept
```

3. Configure a term to reject packets to or from the 192.168/16 prefix.

```
[edit firewall family inet filter filter1]
user@host# set term term2 from address 192.168.0.0/16
user@host# set term term2 then reject
```

4. Configure a term to accept packets destined for either the SSH port or the Telnet port.

```
[edit firewall family inet filter filter1]
user@host# set term term3 from destination-port ssh
user@host# set term term3 from destination-port telnet
user@host# set term term3 then accept
```

5. Configure the last term to reject all packets.

```
[edit firewall family inet filter filter1]
user@host# set term term4 then reject
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input filter1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter1 {
        term term1 {
            from {
                protocol-except [tcp udp];
            }
            then {
                accept;
            }
        }
        term term2 {
            from {
                address 192.168/16;
            }
            then {
                reject;
            }
        }
```

```
        term term3 {
            from {
                destination-port [ssh telnet];
            }
            then {
                accept;
            }
        }
        term term4 {
            then {
                reject;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input filter1;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter filter1` operational mode command.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Match on IPv6 Flags | 1142

Example: Configuring a Filter to Match on Two Unrelated Criteria | 1181

Example: Configuring a Filter to Count Accepted and Rejected Packets

IN THIS SECTION

- Requirements | 1149
- Overview | 1149
- Configuration | 1150
- Verification | 1153

This example shows how to configure a firewall filter to count packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1150

In this example, you use a stateless firewall filter to reject all addresses except 192.168.5.0/24.

Topology

In the first term, the match condition `address 192.168.5.0/24 except` causes this address to be considered a mismatch, and this address is passed to the next term in the filter. The match condition `address 0.0.0.0/0` matches all other packets, and these are counted, logged, and rejected.

In the second term, all packets that passed though the first term (that is, packets whose address matches `192.168.5.0/24`) are counted, logged, and accepted.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1150**
- Configure the Stateless Firewall Filter | **1151**
- Apply the Stateless Firewall Filter to a Logical Interface | **1151**
- Confirm and Commit Your Candidate Configuration | **1152**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter fire1 term 1 from address 192.168.5.0/24 except
set firewall family inet filter fire1 term 1 from address 0.0.0.0/0
set firewall family inet filter fire1 term 1 then count reject_pref1_1
set firewall family inet filter fire1 term 1 then log
set firewall family inet filter fire1 term 1 then reject
set firewall family inet filter fire1 term 2 then count reject_pref1_2
set firewall family inet filter fire1 term 2 then log
set firewall family inet filter fire1 term 2 then accept
set interfaces ge-0/0/1 unit 0 family inet filter input fire1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `fire1`:

1. Create the stateless firewall filter `fire1`.

```
[edit]
user@host# edit firewall family inet filter fire1
```

2. Configure the first term to reject all addresses except those to or from the `192.168.5.0/24` prefix and then count, log, and reject all other packets.

```
[edit firewall family inet filter fire1]
user@host# set term 1 from address 192.168.5.0/24 except
user@host# set term 1 from address 0.0.0.0/0
user@host# set term 1 then count reject_pref1_1
user@host# set term 1 then log
user@host# set term 1 then reject
```

3. Configure the next term to count, log, and accept packets in the `192.168.5.0/24` prefix.

```
[edit firewall family inet filter fire1]
user@host# set term 2 then count reject_pref1_2
user@host# set term 2 then log
user@host# set term 2 then accept
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input fire1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show 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;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input fire1;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter fire1` operational mode command. You can also display the log and individual counters separately by using the following forms of the command:

- `show firewall counter reject_pref1_1`
- `show firewall counter reject_pref1_2`
- `show firewall log`

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Count IP Options Packets | 1158

Example: Configuring a Filter to Count and Discard IP Options Packets | 1154

Example: Configuring a Filter to Count and Discard IP Options Packets

IN THIS SECTION

- Requirements | 1154
- Overview | 1154
- Configuration | 1155
- Verification | 1158

This example shows how to configure a standard stateless firewall to count packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Because the filter term matches on *any* IP option value, the filter term can use the `count` nonterminating action without the `discard` terminating action or (alternatively) without requiring an interface on a 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, or 60-Gigabit Ethernet Enhanced Queuing MPC on an MX Series router.

Overview

In this example, you use a standard stateless firewall filter to count and discard packets that include any IP option value but accept all other packets.

The IP option header field is an optional field in IPv4 headers only. The `ip-options` and `ip-options-except` match conditions are supported for standard stateless firewall filters and service filters only.

NOTE: On M and T series routers, firewall filters cannot count `ip-options` packets on a per option type and per interface basis. A limited work around is to use the `show pfe statistics ip options`

command to see ip-options statistics on a per Packet Forwarding Engine (PFE) basis. See show pfe statistics ip for sample output.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1155
- Configure the Stateless Firewall Filter | 1155
- Apply the Stateless Firewall Filter to a Logical Interface | 1156
- Confirm and Commit Your Candidate Configuration | 1157

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the [edit] hierarchy level.

```
set firewall family inet filter block_ip_options term 10 from ip-options any
set firewall family inet filter block_ip_options term 10 then count option_any
set firewall family inet filter block_ip_options term 10 then discard
set firewall family inet filter block_ip_options term 999 then accept
set interfaces ge-0/0/1 unit 0 family inet filter input block_ip_options
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter:

1. Create the stateless firewall filter `block_ip_options`.

```
[edit]
user@host# edit firewall family inet filter block_ip_options
```

2. Configure the first term to count and discard packets that include any IP options header fields.

```
[edit firewall family inet filter block_ip_options]
user@host# set term 10 from ip-options any
user@host# set term 10 then count option_any
user@host# set term 10 then discard
```

3. Configure the other term to accept all other packets.

```
[edit firewall family inet filter block_ip_options]
user@host# set term 999 then accept
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input block_ip_options
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter block_ip_options {
        term 10 {
            from {
                ip-options any;
            }
            then {
                count option_any;
                discard;
            }
        }
        term 999 {
            then accept;
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
```

```
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input block_ip_options;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter block_ip_options` operational mode command. To display the count of discarded packets separately, enter the `show firewall count option_any` form of the command.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | 778
- Example: Configuring a Filter to Count Accepted and Rejected Packets | 1149
- Example: Configuring a Filter to Count IP Options Packets | 1158

Example: Configuring a Filter to Count IP Options Packets

IN THIS SECTION

- Requirements | 1159
- Overview | 1159
- Configuration | 1159

- Verification | 1165

This example shows how use a stateless firewall filter to count individual IP options packets:

Requirements

This example uses an interface on a 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, or 60-Gigabit Ethernet Enhanced Queuing MPC on an MX Series router. This interface enables you to apply an IPv4 firewall filter (standard or service filter) that can use the `count`, `log`, and `syslog` nonterminating actions on packets that match a *specific* `ip-option` value without having to also use the `discard` terminating action.

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to count IP options packets but not block any traffic. Also, the filter logs packets that have loose or strict source routing.

The IP option header field is an optional field in IPv4 headers only. The `ip-options` and `ip-options-except` match conditions are supported for standard stateless firewall filters and service filters only.

NOTE: On M and T series routers, firewall filters cannot count `ip-options` packets on a per option type and per interface basis. A limited work around is to use the `show pfe statistics ip options` command to see `ip-options` statistics on a per Packet Forwarding Engine (PFE) basis. See show pfe statistics ip for sample output.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1160
- Configure the Stateless Firewall Filter | 1160
- Apply the Stateless Firewall Filter to a Logical Interface | 1162
- Confirm and Commit Your Candidate Configuration | 1163

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter ip_options_filter term match_strict_source from ip-options
strict-source-route
set firewall family inet filter ip_options_filter term match_strict_source then count
strict_source_route
set firewall family inet filter ip_options_filter term match_strict_source then log
set firewall family inet filter ip_options_filter term match_strict_source then accept
set firewall family inet filter ip_options_filter term match_loose_source from ip-options loose-
source-route
set firewall family inet filter ip_options_filter term match_loose_source then count
loose_source_route
set firewall family inet filter ip_options_filter term match_loose_source then log
set firewall family inet filter ip_options_filter term match_loose_source then accept
set firewall family inet filter ip_options_filter term match_record from ip-options record-route
set firewall family inet filter ip_options_filter term match_record then count record_route
set firewall family inet filter ip_options_filter term match_record then accept
set firewall family inet filter ip_options_filter term match_timestamp from ip-options timestamp
set firewall family inet filter ip_options_filter term match_timestamp then count timestamp
set firewall family inet filter ip_options_filter term match_timestamp then accept
set firewall family inet filter ip_options_filter term match_router_alert from ip-options router-
alert
set firewall family inet filter ip_options_filter term match_router_alert then count router_alert
set firewall family inet filter ip_options_filter term match_router_alert then accept
set firewall family inet filter ip_options_filter term match_all then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ip_options_filter
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `ip_option_filter`:

1. Create the stateless firewall filter `ip_option_filter`.

```
[edit]
user@host# edit firewall family inet filter ip_options_filter
```

2. Configure the first term to count, log, and accept packets with the `strict_source_route` IP optional header field.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_strict_source from ip-options strict_source_route
user@host# set term match_strict_source then count strict_source_route
user@host# set term match_strict_source then log
user@host# set term match_strict_source then accept
```

3. Configure the next term to count, log, and accept packets with the `loose-source-route` IP optional header field.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_loose_source from ip-options loose-source-route
user@host# set term match_loose_source then count loose_source_route
user@host# set term match_loose_source then log
user@host# set term match_loose_source then accept
```

4. Configure the next term to count and accept packets with the `record-route` IP optional header field.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_record from ip-options record-route
user@host# set term match_record then count record_route
user@host# set term match_record then accept
```

5. Configure the next term to count and accept packets with the `timestamp` IP optional header field.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_timestamp from ip-options timestamp
user@host# set term match_timestamp then count timestamp
user@host# set term match_timestamp then accept
```

6. Configure the next term to count and accept packets with the `router-alert` IP optional header field.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_router_alert from ip-options router-alert
user@host# set term match_router_alert then count router_alert
user@host# set term match_router_alert then accept
```

7. Create the last term to accept any packet without incrementing any counters.

```
[edit firewall family inet filter ip_option_filter]
user@host# set term match_all then accept
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input ip_options_filter
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter ip_options_filter {
        term match_strict_source {
            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;
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input ip_option_filter;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter ip_option_filter` operational mode command. You can also display the log and individual counters separately by using the following forms of the command:

- `show firewall counter strict_source_route`
- `show firewall counter loose_source_route`
- `show firewall counter record_route`
- `show firewall counter timestamp`
- `show firewall counter router_alert`
- `show firewall log`

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Count Accepted and Rejected Packets | 1149

Example: Configuring a Filter to Count and Discard IP Options Packets | 1154

Example: Configuring a Filter to Count and Sample Accepted Packets

IN THIS SECTION

- Requirements | 1166
- Overview | 1166
- Configuration | 1167
- Verification | 1170

This example shows how to configure a standard stateless firewall filter to count and sample accepted packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin, configure traffic sampling by including the `sampling` statement at the `[edit forwarding-options]` hierarchy level.

Overview

In this example, you use a standard stateless firewall filter to count and sample all packets received on a logical interface.

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.

WARNING: On MX Series routers with MPC3 or MPC4, if firewall filters are configured to count Two-Way Active Measurement Protocol (TWAMP) packets then the count is doubled for all TWAMP packets. There may also be a small increase in round trip time (RTT) when the TWAMP server is hosted on MPC3 or MPC4. This warning does not apply for routers with MPC1 or MPC2 cards.

NOTE: On QFX5130 and QFX5700 when a packet hits multiple firewall filters which has counter action, only the highest priority group counter will increment. For example, assume the packet has hit a IPACL (Port ACL) filter and a IRACL (Routed ACL) filter, both having counter action. According to priority, IRACL has high priority over IPACL. Hence only IRACL counter will increment. If there is no hit in IRACL, then IPACL counter will increment.

Below is the order of priority of user configured ACLs (from highest to lowest),

- Loopback
- IRACL
- IPACL
- IVACL

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1167
- Configure the Stateless Firewall Filter | 1167
- Apply the Stateless Firewall Filter to a Logical Interface | 1168
- Confirm and Commit Your Candidate Configuration | 1169

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter sam term all then count count_sam
set firewall family inet filter sam term all then sample
set interfaces at-2/0/0 unit 301 family inet address 10.1.2.3/30
set interfaces at-2/0/0 unit 301 family inet filter input sam
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `sam`:

1. Create the stateless firewall filter `sam`.

```
[edit]
user@host# edit firewall family inet filter sam
```

2. Configure the term to count and sample all packets.

```
[edit firewall family inet filter sam]
user@host# set term all then count count_sam
user@host# set term all then sample
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input sam
```

NOTE: The Junos OS does not sample packets originating from the router or switch. If you configure a 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.

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter sam {
        term all {
            then {
                count count_sam;
                sample; # default action is accept
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
interfaces {
    at-2/0/0 {
        unit 301 {
            family inet {
                filter {
                    input sam;
                }
                address 10.1.2.3/30;
            }
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

IN THIS SECTION

- Displaying the Packet Counter | 1170
- Displaying the Firewall Filter Log Output | 1170
- Displaying the Sampling Output | 1172

Confirm that the configuration is working properly.

Displaying the Packet Counter

Purpose

Verify that the firewall filter is evaluating packets.

Action

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

Displaying the Firewall Filter Log Output

Purpose

Display the packet header information for all packets evaluated by the firewall filter.

Action

```
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
```

Meaning

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.

Displaying the Sampling Output

Purpose

Verify that the sampling output contains appropriate data.

Action

```
  wtmp.0.gz            Size: 15017, Last changed: Dec 19 13:15:54 wtmp.1.gz            Size:
493, Last changed: Nov 19 13:47:29
  wtmp.2.gz            Size: 57, Last changed: Oct 20 15:24:34
  |                    Pipe through a command
```

```
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
```

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Set the DSCP Bit to Zero | 1172

Example: Configuring a Filter to Set the DSCP Bit to Zero

IN THIS SECTION

- Requirements | 1173
- Overview | 1173
- Configuration | 1173

- Verification | 1176

This example shows how to configure a standard stateless firewall filter based on the Differentiated Services code point (DSCP).

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to match packets on DSCP bit patterns. If the DSCP is `2`, the packet is classified to the `best-effort` forwarding class, and the DSCP is set to `0`. If the DSCP is `3`, the packet is classified to the `best-effort` forwarding class.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1173
- Configure the Stateless Firewall Filter | 1174
- Apply the Stateless Firewall Filter to a Logical Interface | 1174
- Confirm and Commit Your Candidate Configuration | 1175

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall filter filter1 term 1 from dscp 2
set firewall filter filter1 term 1 then forwarding-class best-effort
```

```
set firewall filter filter1 term 1 then dscp 0
set firewall filter filter1 term 2 from dscp 3
set firewall filter filter1 term 2 then forwarding-class best-effort
set interfaces so-0/1/0 unit 0 family inet filter input filter1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `filter1`:

1. Create the stateless firewall filter.

```
[edit]
user@host# edit firewall filter filter1
```

2. Configure the first term to match a packet with a DSCP of `2`, change the DSCP to `0`, and classify the packet to the `best-effort` forwarding class.

```
[edit firewall filter filter1]
user@host# set term 1 from dscp 2
user@host# set term 1 then forwarding-class best-effort
user@host# set term 1 then dscp 0
```

3. Configure the other term to match a packet with a DSCP of `3` and classify the packet to the `best-effort` forwarding class.

```
[edit firewall filter filter1]
user@host# set term 2 from dscp 3
user@host# set term 2 then forwarding-class best-effort
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to the logical interface corresponding to the VPN routing and forwarding (VRF) instance:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces so-0/1/0 unit 0 family inet
```

2. Apply the stateless firewall filter to the logical interface.

```
[ input filter1]
user@host# set filter input filter1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
filter filter1 {
    term term1 {
        from {
            dscp 2;
        }
        then {
            forwarding-class best-effort;
            dscp 0;
        }
    }
    term term2 {
        from {
            dscp 3;
        }
        then {
            forwarding-class best-effort;
        }
```

```
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
so-0/1/0 {
    unit 0 {
        family inet {
            filter input filter1;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the following operational mode commands:

- `show class-of-service`—Displays the entire class-of-service (CoS) configuration, including system-chosen defaults.
- `show class-of-service classifier type dscp`—Displays only the classifiers of the DSCP for IPv4 type.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Count and Sample Accepted Packets | 1165

Example: Configuring a Filter to Set the DSCP Bit to Zero

IN THIS SECTION

- Requirements | 1177
- Overview | 1177
- Configuration | 1177
- Verification | 1180

This example shows how to configure a standard stateless firewall filter based on the Differentiated Services code point (DSCP).

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to match packets on DSCP bit patterns. If the DSCP is **2**, the packet is classified to the **best-effort** forwarding class, and the DSCP is set to **0**. If the DSCP is **3**, the packet is classified to the **best-effort** forwarding class.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1178
- Configure the Stateless Firewall Filter | 1178
- Apply the Stateless Firewall Filter to a Logical Interface | 1179
- Confirm and Commit Your Candidate Configuration | 1179

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall filter filter1 term 1 from dscp 2
set firewall filter filter1 term 1 then forwarding-class best-effort
set firewall filter filter1 term 1 then dscp 0
set firewall filter filter1 term 2 from dscp 3
set firewall filter filter1 term 2 then forwarding-class best-effort
set interfaces ge-0/1/0 unit 0 family inet filter input filter1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **filter1**:

1. Create the stateless firewall filter.

```
[edit]
user@host# edit firewall filter filter1
```

2. Configure the first term to match a packet with a DSCP of **2**, change the DSCP to **0**, and classify the packet to the **best-effort** forwarding class.

```
[edit firewall filter filter1]
user@host# set term 1 from dscp 2
user@host# set term 1 then forwarding-class best-effort
user@host# set term 1 then dscp 0
```

3. Configure the other term to match a packet with a DSCP of **3** and classify the packet to the **best-effort** forwarding class.

```
[edit firewall filter filter1]
user@host# set term 2 from dscp 3
user@host# set term 2 then forwarding-class best-effort
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to the logical interface corresponding to the VPN routing and forwarding (VRF) instance:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/1/0 unit 0 family inet
```

2. Apply the stateless firewall filter to the logical interface.

```
[ input filter1]
user@host# set filter input filter1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
filter filter1 {
    term term1 {
        from {
            dscp 2;
        }
        then {
            forwarding-class best-effort;
            dscp 0;
        }
    }
    term term2 {
```

```
        from {
            dscp 3;
        }
        then {
            forwarding-class best-effort;
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/1/0 {
    unit 0 {
        family inet {
            filter input filter1;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the following operational mode commands:

- **show class-of-service**—Displays the entire class-of-service (CoS) configuration, including system-chosen defaults.
- **show class-of-service classifier type dscp**—Displays only the classifiers of the DSCP for IPv4 type.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Count and Sample Accepted Packets | 1165

Example: Configuring a Filter to Match on Two Unrelated Criteria

IN THIS SECTION

- Requirements | 1181
- Overview | 1181
- Configuration | 1181
- Verification | 1184

This example shows how to configure a standard stateless firewall filter to match on two unrelated criteria.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a standard stateless firewall filter to match IPv4 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.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1182
- Configuring the IPv4 Firewall Filter | 1182
- Applying the IPv4 Firewall Filter to a Logical Interface | 1183

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter ospf_or_131 term protocol_match from protocol ospf
set firewall family inet filter ospf_or_131 term address-match from source-address 10.108.0.0/16
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ospf_or_131
```

Configuring the IPv4 Firewall Filter

Step-by-Step Procedure

To configure the IPv4 firewall filter:

1. Enable configuration of the IPv4 firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter ospf_or_131
   ```

2. Configure the first term to accept OSPF packets.

   ```
   [edit firewall family inet filter ospf_or_131]
   user@host# set term protocol_match from protocol ospf
   ```

   Packets that match the condition are accepted by default. Because another term follows this term, packets that do not match this condition are evaluated by the next term.

3. Configure the second term to accept packets from any IPv4 address in a particular prefix.

   ```
   [edit firewall family inet filter ospf_or_131]
   user@host# set term address_match from source-address 10.108.0.0/16
   ```

   Packets that match this condition are accepted by default. Because this is the last term in the filter, packets that do not match this condition are discarded by default.

Results

Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter ospf_or_131 {
        term protocol_match {
            from {
                protocol ospf;
            }
        }
        term address_match {
            from {
                source-address {
                    10.108.0.0/16;
                }
            }
        }
    }
}
```

Applying the IPv4 Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Enable configuration of a logical interface.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure an IP address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the IPv4 firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input ospf_or_131
```

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input ospf_or_131;
            }
            address 10.1.2.3/30;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

To confirm that the configuration is working properly, enter the `show firewall filter ospf_or_131` operational mode command.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Filter to Match on IPv6 Flags | 1142

Example: Configuring a Filter to Match on Port and Protocol Fields | 1144

Example: Configuring a Filter to Accept DHCP Packets Based on Address

IN THIS SECTION

- Requirements | 1185
- Overview | 1185
- Configuration | 1185
- Verification | 1188

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

Requirements

This example is supported only on MX Series routers and EX Series switches.

Overview

In this example, you create 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`.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1186
- Configure the Stateless Firewall Filter | 1186
- Apply the Firewall Filter to the Loopback Interface | 1187
- Confirm and Commit Your Candidate Configuration | 1187

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter rpf_dhcp term dhcp_term from source-address 0.0.0.0/32
set firewall family inet filter rpf_dhcp term dhcp_term from destination-address
255.255.255.255/32
set firewall family inet filter rpf_dhcp term dhcp_term then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input sam
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter:

1. Create the stateless firewall filter `rpf_dhcp`.

```
[edit]
user@host# edit firewall family inet filter rpf_dhcp
```

2. Configure the term to match 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]
user@host# set term dhcp_term from source-address 0.0.0.0/32
user@host# set term dhcp_term from destination-address 255.255.255.255/32
```

3. Configure the term to accept packets that match the specified conditions.

```
[edit firewall family inet filter rpf_dhcp]
set term dhcp_term then accept
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the filter to the input at the loopback interface:

1. Apply the `rpf_dhcp` filter if packets are not arriving on an expected path.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet rpf-check fail-filter rpf_dhcp
```

2. Configure an address for the loopback interface.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter rpf_dhcp {
        term dhcp_term {
            from {
                source-address {
                    0.0.0.0/32;
                }
                destination-address {
                    255.255.255.255/32;
                }
            }
            then accept;
```

```
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
                rpf-check {
                    fail-filter rpf_dhcp;
                    mode loose;
                }
            }
            address 127.0.0.1/32;
        }
    }
}
```

3. When you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall` operational mode command.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | 778
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | 1065
- Example: Configure a Filter to Block Telnet and SSH Access | 1073

Example: Configuring a Filter to Block TFTP Access | 1086
Example: Configuring a Filter to Accept OSPF Packets from a Prefix | 1189

Example: Configuring a Filter to Accept OSPF Packets from a Prefix

IN THIS SECTION

- Requirements | 1189
- Overview | 1189
- Configuration | 1189
- Verification | 1193

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a filter that accepts only OSPF packets from an address in the prefix 10.108.0.0/16, discarding all other packets with an **administratively-prohibited** ICMP message

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1190
- Configure the Stateless Firewall Filter | 1190
- Apply the Firewall Filter to the Loopback Interface | 1191
- Confirm and Commit Your Candidate Configuration | 1191

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter ospf_filter term term1 from source-address 10.108.0.0/16
set firewall family inet filter ospf_filter term term1 from protocol ospf
set firewall family inet filter ospf_filter term term1 then accept
set firewall family inet filter ospf_filter term default-term then reject administratively-
prohibited
set interfaces lo0 unit 0 family inet address 10.1.2.3/30
set interfaces lo0 unit 0 family inet filter input ospf_filter
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `ospf_filter`:

1. Create the filter.

```
[edit]
user@host# edit firewall family inet filter ospf_filter
```

2. Configure the term that accepts packets.

```
[edit firewall family inet filter ospf_filter]
user@host# set term term1 from source-address 10.108.0.0/16
user@host# set term term1 from protocol ospf
user@host# set term term1 then accept
```

3. Configure the term that rejects all other packets.

```
[edit firewall family inet filter ospf_filter]
user@host# set term default_term then reject administratively-prohibited
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

1. Configure the interface.

```
[edit]
user@host# edit interfaces lo0 unit 0 family inet
```

2. Configure the logical interface IP address.

```
[edit interfaces lo0 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the filter to the input.

```
[edit interfaces lo0 unit 0 family inet]
user@host# set filter input ospf_filter
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
```

```
family inet {
    filter ospf_filter {
        term term1 {
            from {
                source-address {
                    10.108.0.0/16;
                }
                protocol ospf;
            }
            then {
                accept;
            }
        }
        term default_term {
            then {
                reject administratively-prohibited; # default reject action
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
                input ospf_filter;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter ospf_filter` operational mode command.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | **778**
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources | **1065**
- Example: Configure a Filter to Block Telnet and SSH Access | **1073**
- Example: Configuring a Filter to Block TFTP Access | **1086**
- Example: Configuring a Filter to Accept DHCP Packets Based on Address | **1185**

Example: Configuring a Stateless Firewall Filter to Handle Fragments

IN THIS SECTION

- Requirements | **1193**
- Overview | **1194**
- Configuration | **1195**
- Verification | **1199**

This example shows how to create a stateless firewall filter that handles packet fragments.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

IN THIS SECTION

- Topology | 1194

In this example, you create a stateless firewall filter called `fragment-RE` that accepts fragmented packets originating from 10.2.1.0/24 and destined for the BGP port. This example includes the following firewall filter terms:

- `not-from-prefix-term`--Discards packets that are not from 10.2.1.0/24 to ensure that subsequent terms in the firewall filter are matched against packets from 10.2.1.0/24 only.
- `small-offset-term`—Discards small (1–5) offset packets to ensure that subsequent terms in the firewall filter can be matched against all the headers in the packet. In addition, the term adds a record to the system logging destinations for the firewall facility.
- `not-fragmented-term`—Accepts unfragmented TCP packets with a destination port that specifies the BGP protocol. A packet is considered unfragmented if the MF flag is not set and the fragment offset equals 0.
- `first-fragment-term`—Accepts the first fragment of a fragmented TCP packet with a destination port that specifies the BGP protocol.
- `fragment-term`—Accepts all fragments that were not discarded by `small-offset-term`. (packet fragments 6–8191). However, only those fragments that are part of a packet containing a first fragment accepted by `first-fragment-term` are reassembled by the destination device.

Packet fragments offset can be from 1 through 8191.

NOTE: You can move terms within the firewall filter by using the `insert` command. For more information, see "*insert*" in the Junos OS CLI User Guide.

Topology

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1195
- Procedure | 1196

CLI Quick Configuration

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

```
set firewall family inet filter fragment-RE term not-from-prefix-term from source-address
0.0.0.0/0
set firewall family inet filter fragment-RE term not-from-prefix-term from source-address
10.2.1.0/24 except
set firewall family inet filter fragment-RE term not-from-prefix-term then discard
set firewall family inet filter fragment-RE term small-offset-term from fragment-offset 1-5
set firewall family inet filter fragment-RE term small-offset-term then syslog
set firewall family inet filter fragment-RE term small-offset-term then discard
set firewall family inet filter fragment-RE term not-fragmented-term from fragment-offset 0
set firewall family inet filter fragment-RE term not-fragmented-term from fragment-flags "!more-
fragments"
set firewall family inet filter fragment-RE term not-fragmented-term from protocol tcp
set firewall family inet filter fragment-RE term not-fragmented-term from destination-port bgp
set firewall family inet filter fragment-RE term not-fragmented-term then accept
set firewall family inet filter fragment-RE term first-fragment-term from first-fragment
set firewall family inet filter fragment-RE term first-fragment-term from protocol tcp
set firewall family inet filter fragment-RE term first-fragment-term from destination-port bgp
set firewall family inet filter fragment-RE term first-fragment-term then accept
set firewall family inet filter fragment-RE term fragment-term from fragment-offset 6-8191
set firewall family inet filter fragment-RE term fragment-term then accept
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure the stateless firewall filter:

1. Define the stateless firewall filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE
```

2. Configure the first term for the filter.

```
[edit firewall family inet filter fragment-RE ]
user@host# set term not-from-prefix-term from source-address 0.0.0.0/0
user@host# set term not-from-prefix-term from source-address 10.2.1.0/24 except
user@host# set term not-from-prefix-term then discard
```

3. Define the second term for the filter.

```
[edit firewall family inet filter fragment-RE]
user@host# edit term small-offset-term
```

4. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term small-offset-term]
user@host# set from fragment-offset 1-5
```

5. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term small-offset-term]
user@host# set then syslog discard
```

6. Define the third term for the filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE term not-fragmented-term
```

7. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term not-fragmented-term]
user@host# set from fragment-flags "!more-fragments" fragment-offset 0 protocol tcp
destination-port bgp
```

8. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term not-fragmented-term]
user@host# set then accept
```

9. Define the fourth term for the filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE term first-fragment-term
```

10. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term first-fragment-term]
user@host# set from first-fragment protocol tcp destination-port bgp
```

11. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term first-fragment-term]
user@host# set then accept
```

12. Define the last term for the filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE term fragment-term
```

13. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term fragment-term]
user@host# set from fragment-offset 6-8191
```

14. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term fragment-term]
user@host# set then accept
```

Results

Confirm your configuration by entering the `show firewall` command from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
    filter fragment-RE {
        term not-from-prefix-term {
            from {
                source-address {
                    0.0.0.0/0;
                    10.2.1.0/24 except;
                }
            }
            then discard;
        }
        term small-offset-term {
            from {
                fragment-offset 1-5;
            }
            then {
                syslog;
                discard;
            }
        }
        term not-fragmented-term {
            from {
                fragment-offset 0;
```

```
                fragment-flags "!more-fragments";
                protocol tcp;
                destination-port bgp;
            }
            then accept;
        }
        term first-fragment-term {
            from {
                first-fragment;
                protocol tcp;
                destination-port bgp;
            }
            then accept;
        }
        term fragment-term {
            from {
                fragment-offset 6-8191;
            }
            then accept;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Stateless Firewall Filter Configurations | **1200**
- Verifying a Firewall Filter that Handles Fragments | **1200**

To confirm that the configuration is working properly, perform these tasks:

Displaying Stateless Firewall Filter Configurations

Purpose

Verify the configuration of the firewall filter. You can analyze the flow of the filter terms by displaying the entire configuration.

Action

From configuration mode, enter the `show firewall` command.

Meaning

Verify that the output shows the intended configuration of the firewall filter. In addition, verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the `insert` CLI command.

Verifying a Firewall Filter that Handles Fragments

Purpose

Verify that the actions of the firewall filter terms are taken.

Action

Send packets to the device that match the terms.

Meaning

Verify that packets from 10.2.1.0/24 with small fragment offsets are recorded in the device's system logging destinations for the firewall facility.

RELATED DOCUMENTATION

- show route summary

Configuring a Firewall Filter to Prevent or Allow IPv4 Packet Fragmentation

This topic explains how to use the `dont-fragment (set | clear)` actions in an ingress firewall filter to modify the Don't Fragment flag in IPv4 packet headers. These actions are supported only on MPCs in MX Series routers.

You can use a firewall filter on an ingress interface to match IPv4 packets that have the Don't Fragment flag set to one or cleared to zero. Fragmentation is prevented when this flag is set in the packet header. Fragmentation is allowed when the flag is not set.

To prevent an IPv4 packet from being fragmented:

- Configure a filter term that modifies the Don't Fragment flag to one.

```
[edit firewall family inet filter dfSet]
user@host# set term t1 then dont-fragment set
```

To allow an IPv4 packet to be fragmented:

- Configure a filter term that modifies the Don't Fragment flag to zero.

```
[edit firewall family inet filter dfClear]
user@host# set term t1 then dont-fragment clear
```

In the following example, the dfSet firewall filter matches packets that are fragmented and changes the Don't Fragment flag to prevent fragmentation. The dfClear firewall filter matches packets that are not fragmented and changes the Don't Fragment flag to allow fragmentation.

```
[edit firewall family inet]
user@host# edit filter dfSet
user@host# set term t1 from fragment-flags is-fragment
user@host# set term t1 then dont-fragment set
user@host# up
user@host# edit filter dfClear
user@host# set term t1 from fragment-flags dont-fragment
user@host# set term t1 then dont-fragment clear
```

RELATED DOCUMENTATION

Firewall Filter Match Conditions for IPv4 Traffic | 940

Firewall Filter Nonterminating Actions | 871

Stateless Firewall Filter Components | 781

Stateless Firewall Filter Overview | 776

Configuring a Firewall Filter to Discard Ingress IPv6 Packets with a Mobility Extension Header

This topic explains how to configure a firewall filter to discard IPv6 packets that contain a mobility extension header. This feature is supported only on MPCs in MX Series routers.

To configure the stateless firewall filter:

1. Create the stateless firewall filter.

```
[edit]
user@host# edit firewall family inet6 filter filter-name
```

For example:

```
[edit]
user@host# edit firewall family inet6 filter drop-mobility
```

2. Configure a term to discard all traffic that contains a mobility extension header.

```
[edit firewall family inet6 filter drop-mobility]
user@host# set term term1 from extension-header mobility
user@host# set term term1 then discard
```

3. Configure a term to accept all other traffic.

```
[edit firewall family inet6 filter drop-mobility]
user@host# set term term2 then accept
```

4. Apply the firewall filter to a logical interface.

```
[edit interfaces ge-1/2/10 unit 0 family inet6]
user@host# set filter input drop-mobility
```

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Firewall Filter Match Conditions for IPv6 Traffic | 957

Example: Configuring an Egress Filter Based on IPv6 Source or Destination IP Addresses

IN THIS SECTION

- Requirements | 1203
- Overview | 1203
- Configuration | 1204

This example shows how to configure a firewall filter to accept IPv6 packets egressing an `inet6` interface.

Requirements

This topic describes a feature supported on EX4300 and QFX5100 that was introduced in Junos OS Release 19.1R1. No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a typical firewall filter to accept IPv6 source and destination packets in the egress direction of an `inet6` interface. To support filtering in the egress direction, however, you'll first need to set the `set system packet-forwarding-options eracl-ip6-match` using either the `srcip6-and-destip6` or `srcip6-only` option. You'll also need to restart the packet forwarding engine(PFE) after committing the configuration.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1204
- Enable the system for IPv6 address filtering | 1204
- Apply the firewall filter to an egress interface | 1206
- Confirm and Commit Your Candidate Configuration | 1206

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set system packet-forwarding-options eracl-ip6-match srcip6-and-destip6
set firewall family inet6 filter ipv6_filter term t1 from source-address 3001::10/64
set firewall family inet6 filter ipv6_filter term t1 from destination-address 2001::10/64
set interfaces ge-0/0/0 unit 0 family inet6 filter output ipv6_filter
```

Enable the system for IPv6 address filtering

Step-by-Step Procedure

To configure a firewall filter for IPv6 filtering on an `inet6` egress interface:

1. Enable packet forwarding options for matching on either IPv6 source, or IPv6 source and destination IP addresses. In this example, we'll enable both source and destination IP address matching.

```
[edit]
user@host# set system packet-forwarding-options eracl-ip6-match srcip6-and-destip6
```

2. Check, and if appropriate, delete any existing firewall filters that are already bound to the interface you will use for the IPv6 firewall filter:

```
[edit]
user@host# delete interfaces ge-0/0/0 unit 0 family inet6 filter output tcp_filter.
```

3. Commit the changes above, then stop and restart the PFE to accept the `packet-forwarding-options` and clear the PFE for the IPv6 filter(s).

   - For EX4300, use the following:

```
user@host# commit
user@host# run request restart pfe-manager
```

   - For EX4300 virtual chassis, use the following:

```
user@host# commit
user@host# run request system reboot all-members
```

   - For QFX5100, reboot the system:

```
user@host# commit
user@host# run request system reboot
```

4. Create a IPv6 firewall filter named **tcp_filter**.

```
[edit]
user@host# edit firewall family inet6 filter tcp_filter
```

5. Configure the required filter action, here to match packets with an IPv6 source or destination address within the configured range.

```
[edit firewall family inet6 filter tcp_filter]
user@host# set term t1 from source-address 3001::10/64
user@host# set term t1 from destination-address 2001::10/64
```

6. Specify that matched packets are counted, logged to the buffer on the PFE, and accepted.

```
[edit firewall family inet6 filter tcp_filter]
user@host# set term t1 then count egress_ipv6-packets
user@host# set term t1 then log
user@host# set term t1 then accept
```

Apply the firewall filter to an egress interface

Step-by-Step Procedure

To apply the firewall filter to an egress inet6 interface, type the following:

- user@host# **set interfaces ge-0/0/0 unit 0 family inet6 filter output tcp_filter**

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet6 {
    filter tcp_filter {
        term t1 {
            from {
                source-address 3001::10/64;
                destination-address 2001::10/64;
```

```
            }
            then {
                count egress_ipv6-packets;
                log;
                accept;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet6 {
            filter {
                output tcp_filter;
            }
             source-address 3001::10/64;
             destination-address 2001::10/64;
        }
    }
}
```

3. When you are done configuring the device, commit the candidate configuration.

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

RELATED DOCUMENTATION

eracl-ip6-match

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Rate-Limiting Filter Based on Destination Class

IN THIS SECTION

- Requirements | 1208
- Overview | 1208
- Configuration | 1208
- Verification | 1212

This example shows how to configure a rate-limiting stateless firewall filter.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin, configure the destination class `class1`.

Overview

In this example, you use a stateless firewall filter to set rate limits based on a destination class.

To activate a policer from within a stateless firewall filter configuration:

- Create a template for the policer by including the `policer` *`policer-name`* statement.
- Reference the policer in a filter term that specifies the policer in the `policer` *`policer-name`* nonterminating action.

You can also activate a policer by including the `policer (input | output)` *`policer-template-name`* statement at a logical interface.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1209
- Configure the Stateless Firewall Filter | 1209
- Apply the Stateless Firewall Filter to a Logical Interface | 1210

- Confirm and Commit Your Candidate Configuration | 1210

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall policer police_class1 if-exceeding bandwidth-limit 25m
set firewall policer police_class1 if-exceeding burst-size-limit 25m
set firewall policer police_class1 then discard
set firewall filter rl_dclass1 term term1 from destination-class class1
set firewall filter rl_dclass1 term term1 then policer police_class1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.1/30
set interfaces ge-0/0/1 unit 0 family inet filter input rl_dclass1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter `rl_dclass1` with policer `police_class1` for destination class `class1`:

1. Create the policer template `police_class1`.

```
[edit]
user@host# edit firewall policer police_class1
```

2. Configure the policer template `police_class1`.

```
[edit firewall policer police_class1]
user@host# set if-exceeding bandwidth-limit 25m
user@host# set if-exceeding burst-size-limit 25m
user@host# set then discard
```

3. Create the stateless firewall filter `rl_dclass1`.

```
[edit]
user@host# edit firewall filter rl_dclass1
```

4. Configure a filter term that uses policer `police_class1` to rate-limit traffic for destination class `class1`.

```
[edit firewall filter rl_dclass1]
user@host# set term term1 from destination-class class1
user@host# set term term1 then policer police_class1
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the filter `rl_dclass1` to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.1/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input rl_dclass1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
policer police_class1 {
    if-exceeding {
        bandwidth-limit 25m;
        burst-size-limit 25m;
    }
    then discard;
}
filter rl_dclass1 {
    term term1 {
        from {
            destination-class class1;
        }
        then policer police_class1;
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input rl_dclass1;
            }
            address 10.1.2.1/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show class-of-service ge-0/0/1` operational mode command.

RELATED DOCUMENTATION

- Understanding How to Use Standard Firewall Filters | 778
- Filtering Packets Received on an Interface Set Overview | 1374
- Example: Filtering Packets Received on an Interface Set | 1358

CHAPTER 17

Configuring Firewall Filters in Logical Systems

IN THIS CHAPTER

- Firewall Filters in Logical Systems Overview | **1213**
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | **1215**
- References from a Firewall Filter in a Logical System to Subordinate Objects | **1218**
- References from a Firewall Filter in a Logical System to Nonfirewall Objects | **1220**
- References from a Nonfirewall Object in a Logical System to a Firewall Filter | **1223**
- Example: Configuring Filter-Based Forwarding | **1230**
- Example: Configuring Filter-Based Forwarding on Logical Systems | **1236**
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | **1250**
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | **1256**
- Unsupported Firewall Filter Statements for Logical Systems | **1262**
- Unsupported Actions for Firewall Filters in Logical Systems | **1265**
- Filter-Based Forwarding for Routing Instances | **1272**
- Forwarding Table Filters for Routing Instances on ACX Series Routers | **1273**
- Configuring Forwarding Table Filters | **1274**

Firewall Filters in Logical Systems Overview

IN THIS SECTION

- Logical Systems | **1214**
- Firewall Filters in Logical Systems | **1214**
- Identifiers for Firewall Objects in Logical Systems | **1214**

Logical Systems

With the Junos OS, you can partition a single physical router or switch into multiple logical devices that perform independent routing tasks. Because logical systems perform a subset of the tasks once handled by the physical router or switch, logical systems offer an effective way to maximize the use of a single router or switch.

Firewall Filters in Logical Systems

You can configure a separate set of firewall filters for each logical system on a router or switch. To configure a filter in a logical system, you must define the filter in the `firewall` stanza at the `[edit logical-systems` *`logical-system-name`*`]` hierarchy level, and you must apply the filter to a *logical interface* that is also configured at the `[edit logical-systems` *`logical-system-name`*`]` hierarchy level.

Identifiers for Firewall Objects in Logical Systems

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 `__ls1/` as the prefix.

RELATED DOCUMENTATION

- Stateless Firewall Filter Types
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | **1215**
- Unsupported Firewall Filter Statements for Logical Systems | **1262**
- Unsupported Actions for Firewall Filters in Logical Systems | **1265**
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | **1250**
- Introduction to Logical Systems
- Logical Systems Operations and Restrictions

Guidelines for Configuring and Applying Firewall Filters in Logical Systems

IN THIS SECTION

- Statement Hierarchy for Configuring Firewall Filters in Logical Systems | 1215
- Filter Types in Logical Systems | 1216
- Firewall Filter Protocol Families in Logical Systems | 1216
- Firewall Filter Match Conditions in Logical Systems | 1217
- Firewall Filter Actions in Logical Systems | 1217
- Statement Hierarchy for Applying Firewall Filters in Logical Systems | 1217

Statement Hierarchy for Configuring Firewall Filters in Logical Systems

To configure a *firewall filter* in a logical system, include the `filter`, `service-filter`, or `simple-filter` statement at the `[edit logical-systems` *`logical-system-name`* `firewall family` *`family-name`*`]` hierarchy level.

```
[edit]
logical systems {
    logical-system-name {
        firewall {
            family family-name {
                filter filter-name {
                    interface-specific;
                    physical-interface-filter;
                    term term-name {
                        filter filter-name;
                        from {
                            match-conditions;
                        }
                        then {
                            actions;
                        }
                    }
                }
                service-filter filter-name { # For 'family inet' or 'family inet6' only.
```

```
                    term term-name {
                        from {
                            match-conditions;
                        }
                        then {
                            actions;
                        }
                    }
                }
                simple-filter filter-name { # For 'family inet' only.
                    term term-name {
                        from {
                            match-conditions;
                        }
                        then {
                            actions;
                        }
                    }
                }
            }
        }
    }
}
```

Filter Types in Logical Systems

There are no special restrictions on the types of stateless firewall filter types that you can configure in logical systems.

In a logical system, you can use the same types of stateless firewall filters that are available on a physical router or switch:

- Standard stateless firewall filters
- Service filters
- Simple filters

Firewall Filter Protocol Families in Logical Systems

There are no special restrictions on the protocol families supported with stateless firewall filters in logical systems.

In a logical system, you can filter the same protocol families as you can on a physical router or switch.

- Standard stateless firewall filters—In logical systems, you can filter the following traffic types: protocol-independent, IPv4, IPv6, MPLS, MPLS-tagged IPv4 or IPv6, VPLS, Layer 2 circuit cross-connection, and Layer 2 bridging.
- Service filters—In logical systems, you can filter IPv4 and IPv6 traffic.
- Simple filters—In logical systems, you can filter IPv4 traffic only.

Firewall Filter Match Conditions in Logical Systems

There are no special restrictions on the match conditions supported with stateless firewall filters in logical systems.

Firewall Filter Actions in Logical Systems

There are no special restrictions on the actions supported with stateless firewall filters in logical systems.

Statement Hierarchy for Applying Firewall Filters in Logical Systems

To apply a firewall filter in a logical system, include the `filter` *`filter-name`*, `service-filter` *`service-filter-name`*, or `simple-filter` *`simple-filter-name`* statement to a *logical interface* in the logical system.

The following configuration shows the hierarchy levels at which you can apply the statements:

```
[edit]
logical-systems logical-system-name {
    interfaces {
        interface-name {
            unit logical-unit-number {
                family family-name {
                    filter {
                        group group-name;
                        input filter-name;
                        input-list [ filter-names ];
                        output filter-name;
                        output-list [ filter-names ]
                    }
                    rpf-check { # For 'family inet' or 'family inet6' only.
                        fail-filter filter-name;
                        mode loose;
                    }
```

```
                    service { # For 'family inet' or 'family inet6' only.
                        input {
                            service-set service-set-name <service-filter service-filter-name>;
                            post-service-filter service-filter-name;
                        }
                        output {
                            service-set service-set-name <service-filter service-filter-name>;
                        }
                    }
                    simple-filter { # For 'family inet' only.
                        input simple-filter-name;
                    }
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

- Firewall Filters in Logical Systems Overview | 1213
- References from a Firewall Filter in a Logical System to Subordinate Objects | 1218
- References from a Firewall Filter in a Logical System to Nonfirewall Objects | 1220
- References from a Nonfirewall Object in a Logical System to a Firewall Filter | 1223
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods | 1250
- Unsupported Firewall Filter Statements for Logical Systems | 1262
- Unsupported Actions for Firewall Filters in Logical Systems | 1265

References from a Firewall Filter in a Logical System to Subordinate Objects

IN THIS SECTION

- Resolution of References from a Firewall Filter to Subordinate Objects | 1219

- Valid Reference from a Firewall Filter to a Subordinate Object | 1219

Resolution of References from a Firewall Filter to Subordinate Objects

If a *firewall filter* defined in a logical system references a subordinate object (for example, a policer or prefix list), that subordinate object must be defined within the `firewall` stanza of the same logical system. For example, if a firewall filter configuration references a policer, the firewall filter and the policer must be configured under the same `[edit logical-systems` *`logical-system-name`* `firewall]` hierarchy level.

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.

Valid Reference from a Firewall Filter to a Subordinate Object

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

```
[edit]
logical systems {
    ls-A {
        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;
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

- Firewall Filters in Logical Systems Overview | 1213
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | 1215
- References from a Firewall Filter in a Logical System to Nonfirewall Objects | 1220
- References from a Nonfirewall Object in a Logical System to a Firewall Filter | 1223

References from a Firewall Filter in a Logical System to Nonfirewall Objects

IN THIS SECTION

- Resolution of References from a Firewall Filter to Nonfirewall Objects | 1220
- Valid Reference to a Nonfirewall Object Outside of the Logical System | 1221

Resolution of References from a Firewall Filter to 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.

Valid Reference to a Nonfirewall Object Outside of the Logical System

This example configuration illustrates an exception to the general rule that the objects referenced by a *firewall filter* in a logical system must be defined under the same logical system as the referencing object.

In the following scenario, the service filter `inetsf1` is applied to IPv4 traffic associated with the service set `fred` at the *logical interface* `fe-0/3/2.0`, which is on an adaptive services interface.

- Service filter `inetsf1` is defined in `ls-B` and references prefix list `prefix1`.
- Service set `fred` is defined at the main services hierarchy level, and the policy framework software searches the `[edit services]` hierarchy for the definition of the `fred` service set.

Because service rules cannot be configured in logical systems. firewall filter configurations in the `[edit logical-systems logical-system logical-system-name]` hierarchy are allowed to reference *service sets* outside the logical system hierarchy.

```
[edit]
logical-systems {
    ls-B {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        service {
                            input {
                                service-set fred service-filter inetsf1;
                            }
                        }
                    }
                }
            }
        }
        policy-options {
            prefix-list prefix1 {
                1.1.0.0/16;
                1.2.0.0/16;
                1.3.0.0/16;
            }
        }
        firewall { # Under logical-system 'ls-B'.
            family inet {
```

```
                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;
                    }
                }
                service-filter inetsf1 {
                    term term1 {
                        from {
                            source-prefix-list {
                                prefix1;
                            }
                        }
                        then count prefix1;
                    }
                }
            }
            policer pol1 {
                if-exceeding {
                    bandwidth-limit 401k;
                    burst-size-limit 50k;
                }
                then discard;
            }
        }
    }
} # End of logical systems configuration.
services { # Main services hierarchy level.
    service-set fred {
```

```
        max-flows 100;
        interface-service {
            service-interface sp-1/2/0.0;
        }
    }
}
```

RELATED DOCUMENTATION

- Firewall Filters in Logical Systems Overview | 1213
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | 1215
- References from a Firewall Filter in a Logical System to Subordinate Objects | 1218
- References from a Nonfirewall Object in a Logical System to a Firewall Filter | 1223

References from a Nonfirewall Object in a Logical System to a Firewall Filter

IN THIS SECTION

- Resolution of References from a Nonfirewall Object to a Firewall Filter | 1223
- Invalid Reference to a Firewall Filter Outside of the Logical System | 1224
- Valid Reference to a Firewall Filter Within the Logical System | 1225
- Valid Reference to a Firewall Filter Outside of the Logical System | 1228

Resolution of References from a Nonfirewall Object to a Firewall Filter

If a nonfirewall filter object in a logical system references an object in a *firewall filter* configured in a logical system, the reference is resolved using the following logic:

- If the nonfirewall filter object is configured in a logical system that includes firewall filter configuration statements, the policy framework software searches the `[edit logical-systems` *`logical-system-name`* `firewall]` hierarchy level. Firewall filter configurations that belong to *other* logical systems or to the main `[edit firewall]` hierarchy level are not searched.

- If the nonfirewall filter object is configured in a logical system that does not include any firewall filter configuration statements, the policy framework software searches the firewall configurations defined at the `[edit firewall]` hierarchy level.

Invalid Reference to a Firewall Filter Outside of the Logical System

This example configuration illustrates an unresolvable reference from a nonfirewall object in a logical system to a firewall filter.

In the following scenario, the stateless firewall filters `filter1` and `fred` are applied to the *logical interface* `fe-0/3/2.0` in the logical system `ls-C`.

- Filter `filter1` is defined in `ls-C`.
- Filter `fred` is defined in the main firewall configuration.

Because `ls-C` contains firewall filter statements (for `filter1`), the policy framework software resolves references to and from firewall filters by searching the `[edit logical systems ls-C firewall]` hierarchy level. Consequently, the reference from `fe-0/3/2.0` in the logical system to `fred` in the main firewall configuration cannot be resolved.

```
[edit]
logical-systems {
    ls-C {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        filter {
                            input-list [ filter1 fred ];
                        }
                    }
                }
            }
        }
        firewall { # Under logical system 'ls-C'.
            family inet {
                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;
                }
            }
        }
        policer pol1 {
            if-exceeding {
                bandwidth-limit 401k;
                burst-size-limit 50k;
            }
            then discard;
        }
    }
}
} # End of logical systems
firewall { # Under the main firewall hierarchy level
    family inet {
        filter fred {
            term one {
                from {
                    source-address 11.1.0.0/16;
                }
                then {
                    log;
                    reject host-unknown;
                }
            }
        }
    }
} # End of main firewall configurations.
```

Valid Reference to a Firewall Filter Within the Logical System

This example configuration illustrates resolvable references from a nonfirewall object in a logical system to two firewall filter.

In the following scenario, the stateless firewall filters `filter1` and `fred` are applied to the logical interface `fe-0/3/2.0` in the logical system `ls-C`.

- Filter `filter1` is defined in `ls-C`.
- Filter `fred` is defined in `ls-C` and also in the main firewall configuration.

Because `ls-C` contains firewall filter statements, the policy framework software resolves references to and from firewall filters by searching the `[edit logical systems ls-C firewall]` hierarchy level. Consequently, the references from `fe-0/3/2.0` in the logical system to `filter1` and `fred` use the stateless firewall filters configured in `ls-C`.

```
[edit]
logical-systems {
    ls-C {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        filter {
                            input-list [ filter1 fred ];
                        }
                    }
                }
            }
        }
        firewall { # Under logical system 'ls-C'.
            family inet {
                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;
                    }
```

```
                }
                filter fred { # This 'fred' is in 'ls-C'.
                    term one {
                        from {
                            source-address 10.1.0.0/16;
                        }
                        then {
                            log;
                            reject host-unknown;
                        }
                    }
                }
            }
            policer pol1 {
                if-exceeding {
                    bandwidth-limit 401k;
                    burst-size-limit 50k;
                }
                then discard;
            }
        }
    }
} # End of logical systems configurations.
firewall { # Main firewall filter hierarchy level
    family inet {
        filter fred {
            term one {
                from {
                    source-address 11.1.0.0/16;
                }
                then {
                    log;
                    reject host-unknown;
                }
            }
        }
    }
} # End of main firewall configurations.
```

Valid Reference to a Firewall Filter Outside of the Logical System

This example configuration illustrates resolvable references from a nonfirewall object in a logical system to two firewall filter.

In the following scenario, the stateless firewall filters `filter1` and `fred` are applied to the logical interface `fe-0/3/2.0` in the logical system `ls-C`.

- Filter `filter1` is defined in the main firewall configuration.
- Filter `fred` is defined in the main firewall configuration.

Because `ls-C` does not contain any firewall filter statements, the policy framework software resolves references to and from firewall filters by searching the `[edit firewall]` hierarchy level. Consequently, the references from `fe-0/3/2.0` in the logical system to `filter1` and `fred` use the stateless firewall filters configured in the main firewall configuration.

```
[edit]
logical-systems {
    ls-C {
        interfaces {
            fe-0/3/2 {
                unit 0 {
                    family inet {
                        filter {
                            input-list [ filter1 fred ];
                        }
                    }
                }
            }
        }
    }
} # End of logical systems configurations.
firewall { # Main firewall hierarchy level.
    family inet {
        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;
            }
        }
        filter fred {
            term one {
                from {
                    source-address 11.1.0.0/16;
                }
                then {
                    log;
                    reject host-unknown;
                }
            }
        }
    }
    policer pol1 {
        if-exceeding {
            bandwidth-limit 701k;
            burst-size-limit 70k;
        }
        then discard;
    }
} # End of main firewall configurations.
```

RELATED DOCUMENTATION

- Firewall Filters in Logical Systems Overview | 1213
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | 1215
- References from a Firewall Filter in a Logical System to Subordinate Objects | 1218
- References from a Firewall Filter in a Logical System to Nonfirewall Objects | 1220

Example: Configuring Filter-Based Forwarding

IN THIS SECTION

- Requirements | 1230
- Overview | 1230
- Configuration | 1231

Filter-based forwarding (FBF), which is also called Policy Based Routing (PBR), provides a a simple but powerful way to route IP traffic to different interfaces on the basis of Layer-3 or Layer-4 parameters.

FBF works by using match conditions in a firewall filter to select certain traffic and then direct it to a given routing instance that points to the desired next hop. To ensure the next hop is resolvable, interface routes from the main routing table are shared via RIB group with the routing table(s) specified in the routing instance(s).

Match conditions can include the source or destination IP address, source or destination port, IP protocol, DSCP value, TCP flag, ICMP type, and packet length.

Requirements

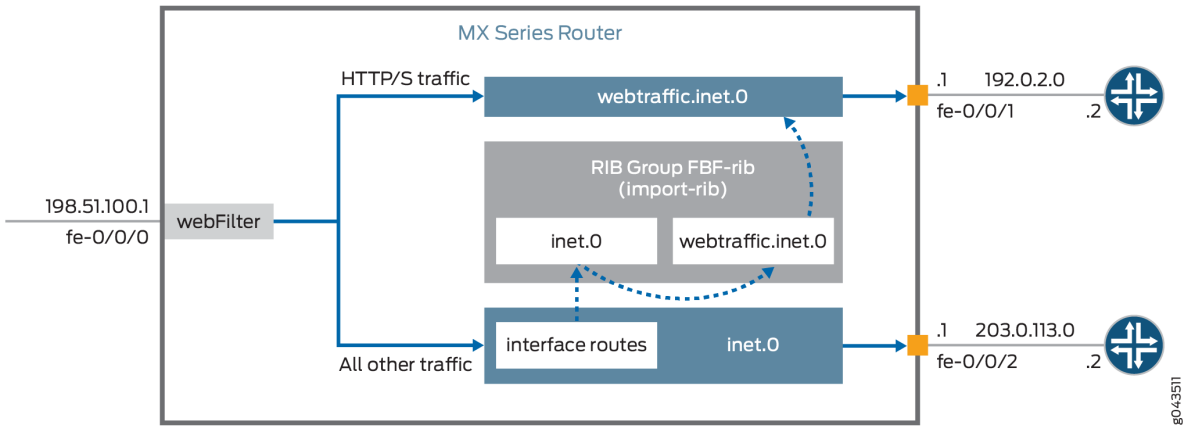
This example has the following hardware and software requirements:

- MX Series 5G Universal Routing Platform as the routing device with the firewall filter configured.
- Junos OS Release 13.3 or later running on the routing device with the firewall filter configured.

Overview

This example shows the configuration settings you need to set up filter-based forwarding on a single device. Figure 54 on page 1231 shows the ingress and egress interfaces on an MX Series router and illustrates the logical flow of events as packets traverse the device.

Figure 54: Filter-Based Forwarding to Specified Interfaces



A firewall filter called **webFilter** is attached to the ingress interface, **fe-0/0/0**. Packets arriving over the interface are evaluated against the match conditions specified in the filter, the logic of which directs HTTP and HTTPS traffic to a routing instance called **webtraffic**. This routing instance accomplishes three things: first, it establishes a routing table called **webtraffic.inet.0**; second, it lets you define a static route and next hop; and third, lets you configure the instance for forwarding traffic to the next hop (here, 192.0.2.2 on interface **fe-0/0/1**).

Term 2 in the firewall filter, **then accept**, specifies that all non-matching traffic take a different path. We define a static route with next hop of 203.0.113.2 to have this traffic egress the device via **fe-0/0/2**. The route is automatically installed in the master routing table, **inet.0**.

The last (logical) step in setting up FBF is to ensure that both routes are resolvable. The RIB group (**FBF-rib** in this example) makes it so interface-routes from **inet.0** can be shared with **webtraffic.inet.0**.

For examples that focus on a specific use case or multi-device topologies, see the Related Topics.

Configuration

IN THIS SECTION

- Procedure | 1232

Procedure

CLI Quick Configuration

Both copy-paste and step-by-step instructions for creating filter-based forwarding on a single device are provided.

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

Configure a device for filter-based forwarding

```
set interfaces fe-0/0/0 unit 0 family inet address 198.51.100.1/24
set interfaces fe-0/0/0 unit 0 family inet filter input webFilter
set interfaces fe-0/0/1 unit 0 family inet address 192.0.2.1/24
set interfaces fe-0/0/2 unit 0 family inet address 203.0.113.1/24
set firewall family inet filter webFilter term 1 from destination-port http
set firewall family inet filter webFilter term 1 from destination-port https
set firewall family inet filter webFilter term 1 then routing-instance webtraffic
set firewall family inet filter webFilter term 2 then accept
set routing-instances webtraffic routing-options static route 0.0.0.0/0 next-hop 192.0.2.2
set routing-instances webtraffic instance-type forwarding
set routing-options static route 0.0.0.0/0 next-hop 203.0.113.2
set routing-options rib-groups FBF-rib import-rib inet.0
set routing-options rib-groups FBF-rib import-rib webtraffic.inet.0
set routing-options interface-routes rib-group inet FBF-rib
```

Step-by-Step Procedure

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

To configure the device:

1. Configure the inbound interface and attach the **webFilter** firewall filter to it.

```
[edit interfaces fe-0/0/0 unit 0 family inet]
user@device# set filter input webFilter
user@device# set address 198.51.100.1/24
```

2. Configure the outbound interfaces, one for Web traffic and the other for all other traffic.

```
[edit interfaces]
user@device# set fe-0/0/1 unit 0 family inet address 192.0.2.1/24
user@device# set fe-0/0/2 unit 0 family inet address 203.0.113.1/24
```

3. Configure the firewall filter to pass Web traffic to the **webtraffic** routing instance and all other traffic to 203.0.113.1.

```
[edit firewall family inet filter webFilter]
user@device# set term 1 from destination-port http
user@device# set term 1 from destination-port https
user@device# set term 1 then routing-instance webtraffic
user@device# set term 2 then accept
```

4. Optional: Monitor traffic handling of the firewall filter by adding a counter>

```
[edit interfaces fe-0/0/0 unit 0 family inet]
user@device# set firewall family inet filter webFilter term 1 then count webtraffic-count
```

5. Create the **webtraffic** routing instance and configure it to forward Web traffic to fe-0/0/1.

```
[edit routing-instances webtraffic]
user@device# set routing-options static route 0.0.0.0/0 next-hop 192.0.2.2
user@device# set instance-type forwarding
```

6. Create a route for non-Web traffic (the route is automatically installed in the inet.0 routing table).

```
[edit routing-options]
user@device# set static route 0.0.0.0/0 next-hop 203.0.113.2
```

7. Create a RIB group called **FBF-rib**, and configure it so **inet.0** shares interface routes with **webtraffic.inet.0**, and then associate a routing table group with the routing device's interfaces, and specify routing table groups into which interface routes are imported..

```
[edit routing-options]
user@device# set rib-groups FBF-rib import-rib inet.0
user@device# set rib-groups FBF-rib import-rib webtraffic.inet.0
```

8. Associate a routing table group with the routing device's interfaces, and specify routing table groups into which interface routes are imported.

```
[edit routing-options]
user@device# set interface-routes rib-group inet FBF-rib
```

Results

From configuration mode, confirm your configuration by entering the `show firewall`, `show routing-instances`, `show routing-options`, and `show interfaces`, commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

If you are done configuring the device, enter `commit` from configuration mode.

```
user@device# show interfaces fe-0/0/0
unit 0 {
    family inet {
        filter {
            input webFilter;
        }
        address 198.51.100.1/24;
    }
}
user@device# show interfaces fe-0/0/1
unit 0 {
    family inet {
        address 192.0.2.1/24;
    }
}
user@device# show interfaces fe-0/0/2
unit 0 {
    family inet {
```

```
        address 203.0.113.1/24;
    }
}
user@device# show firewall
family inet {
    filter webFilter {
        term 1 {
            from {
                destination-port [ http https ];
            }
            then {
                routing-instance webtraffic;
            }
        }
        term 2 {
            then accept;
        }
    }
}
```

```
user@device# show routing-options
interface-routes {
    rib-group inet FBF-rib;
}
static {
    route 0.0.0.0/0 next-hop 203.0.113.2;
}
rib-groups {
    FBF-rib {
        import-rib [ inet.0 webtraffic.inet.0 ];
    }
}
```

```
user@device# show routing-instances
webtraffic {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 192.0.2.2;
        }
```

```
    }
}
```

RELATED DOCUMENTATION

- Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**
- Configuring Filter-Based Forwarding
- Understanding Filter-Based Forwarding | **1854**
- *Using Filter-Based Forwarding to Select Traffic to Be Secured*
- Example: Configuring Filter-Based Forwarding on the Source Address | **1497**
- Understanding RIB Groups

Example: Configuring Filter-Based Forwarding on Logical Systems

IN THIS SECTION

- Requirements | **1236**
- Overview | **1237**
- Configuration | **1240**
- Verification | **1247**

This example shows how to configure filter-based forwarding within a logical system. The filter classifies packets to determine their forwarding path within the ingress routing device.

Requirements

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

Overview

IN THIS SECTION

- Topology | 1239

Filter-based forwarding is supported for IP version 4 (IPv4) and IP version 6 (IPv6).

Use filter-based forwarding for service provider selection when customers have Internet connectivity provided by different ISPs yet share a common access layer. When a shared media (such as a cable modem) is used, a mechanism on the common access layer looks at Layer 2 or Layer 3 addresses and distinguishes between customers. You can use filter-based forwarding when the common access layer is implemented using a combination of Layer 2 switches and a single router.

With filter-based forwarding, all packets received on an interface are considered. Each packet passes through a filter that has match conditions. If the match conditions are met for a filter and you have created a routing instance, filter-based forwarding is applied to a packet. The packet is forwarded based on the next hop specified in the routing instance. For static routes, the next hop can be a specific LSP.

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

To configure filter-based forwarding, perform the following tasks:

- Create a match filter on an ingress router or switch. To specify a match filter, include the `filter` *`filter-name`* statement at the `[edit firewall]` hierarchy level. A packet that passes through the filter is compared against a set of rules to classify it and to determine its membership in a set. Once classified, the packet is forwarded to a routing table specified in the accept action in the filter description language. The routing table then forwards the packet to the next hop that corresponds to the destination address entry in the table.
- Create routing instances that specify the routing table(s) to which a packet is forwarded, and the destination to which the packet is forwarded at the `[edit routing-instances]` or `[edit logical-systems` *`logical-system-name`* `routing-instances]` hierarchy level. For example:

```
[edit]
routing-instances {
    routing-table-name1 {
        instance-type forwarding;
```

```
        routing-options {
            static {
                route 0.0.0.0/0 nexthop 10.0.0.1;
            }
        }
    }
    routing-table-name2 {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 nexthop 10.0.0.2;
            }
        }
    }
}
```

- Create a routing table group that adds interface routes to the forwarding routing instances used in filter-based forwarding (FBF), as well as to the default routing instance `inet.0`. This part of the configuration resolves the routes installed in the routing instances to directly connected next hops on that interface. Create the routing table group at the `[edit routing-options]` or `[edit logical-systems` *`logical-system-name`* `routing-options]` hierarchy level.

NOTE: Specify `inet.0` as one of the routing instances that the interface routes are imported into. If the default instance `inet.0` is not specified, interface routes are not imported into the default routing instance.

This example shows a packet filter that directs customer traffic to a next-hop router in the domains, SP 1 or SP 2, based on the packet's source address.

If the packet has a source address assigned to an SP 1 customer, destination-based forwarding occurs using the sp1-route-table.inet.0 routing table. If the packet has a source address assigned to an SP 2 customer, destination-based forwarding occurs using the sp2-route-table.inet.0 routing table. If a packet does not match either of these conditions, the filter accepts the packet, and destination-based forwarding occurs using the standard inet.0 routing table.

One way to make filter-based forwarding work within a logical system is to configure the firewall filter on the logical system that receives the packets. Another way is to configure the firewall filter on the main router and then reference the logical system in the firewall filter. This example uses the second approach. The specific routing instances are configured within the logical system. Because each routing

instance has its own routing table, you have to reference the routing instances in the firewall filter, as well. The syntax looks as follows:

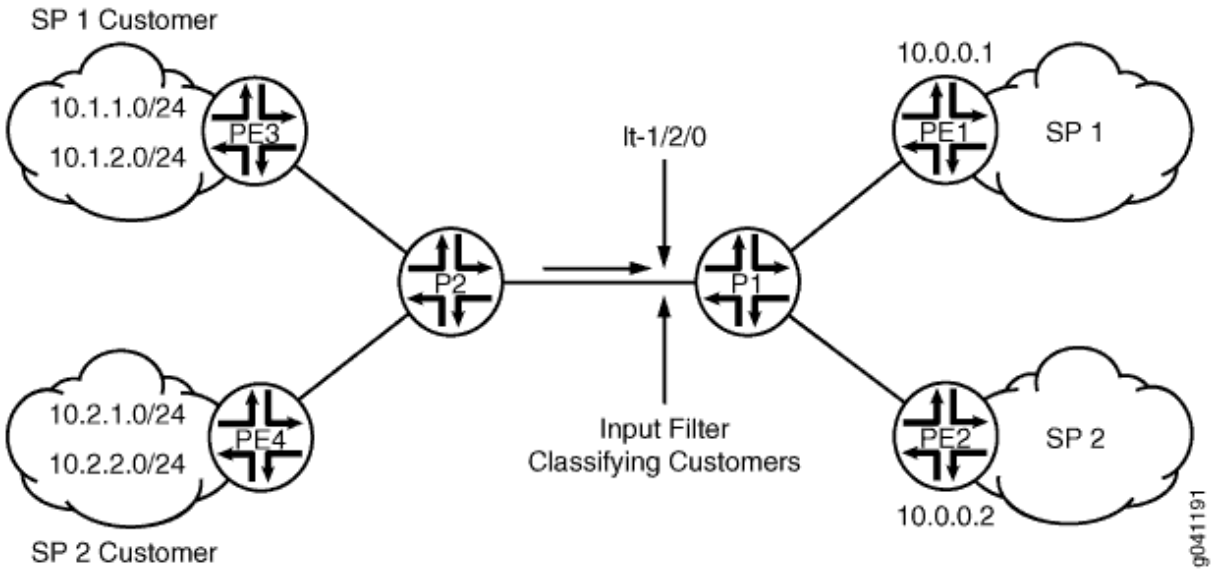
```
[edit firewall filter filter-name term term-name]
user@host# set then logical-system logical-system-name routing-instance routing-instance-name
```

Topology

Figure 55 on page 1239 shows the topology used in this example.

On Logical System P1, an input filter classifies packets received from Logical System PE3 and Logical System PE4. The packets are routed based on the source addresses. Packets with source addresses in the 10.1.1.0/24 and 10.1.2.0/24 networks are routed to Logical System PE1. Packets with source addresses in the 10.2.1.0/24 and 10.2.2.0/24 networks are routed to Logical System PE2.

Figure 55: Logical Systems with Filter-Based Forwarding



To establish connectivity, OSPF is configured on all of the interfaces. For demonstration purposes, loopback interface addresses are configured on the routing devices to represent networks in the clouds.

The "CLI Quick Configuration" on page 1240 section shows the entire configuration for all of the devices in the topology. The "Configuring the Routing Instances on the Logical System P1" on page 1243 and "Configuring the Firewall Filter on the Main Router" on page 1242 sections shows the step-by-step configuration of the ingress routing device, Logical System P1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1240
- Configuring the Firewall Filter on the Main Router | 1242
- Configuring the Routing Instances on the Logical System P1 | 1243
- Results | 1245

CLI Quick Configuration

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

```
set firewall filter classify-customers term sp1-customers from source-address 10.1.1.0/24
set firewall filter classify-customers term sp1-customers from source-address 10.1.2.0/24
set firewall filter classify-customers term sp1-customers then log
set firewall filter classify-customers term sp1-customers then logical-system P1 routing-
instance sp1-route-table
set firewall filter classify-customers term sp2-customers from source-address 10.2.1.0/24
set firewall filter classify-customers term sp2-customers from source-address 10.2.2.0/24
set firewall filter classify-customers term sp2-customers then log
set firewall filter classify-customers term sp2-customers then logical-system P1 routing-
instance sp2-route-table
set firewall filter classify-customers term default then accept
set logical-systems P1 interfaces lt-1/2/0 unit 10 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 10 peer-unit 9
set logical-systems P1 interfaces lt-1/2/0 unit 10 family inet filter input classify-customers
set logical-systems P1 interfaces lt-1/2/0 unit 10 family inet address 172.16.0.10/30
set logical-systems P1 interfaces lt-1/2/0 unit 13 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 13 peer-unit 14
set logical-systems P1 interfaces lt-1/2/0 unit 13 family inet address 172.16.0.13/30
set logical-systems P1 interfaces lt-1/2/0 unit 17 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 17 peer-unit 18
set logical-systems P1 interfaces lt-1/2/0 unit 17 family inet address 172.16.0.17/30
set logical-systems P1 protocols ospf rib-group fbf-group
set logical-systems P1 protocols ospf area 0.0.0.0 interface all
```

```
set logical-systems P1 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems P1 routing-instances sp1-route-table instance-type forwarding
set logical-systems P1 routing-instances sp1-route-table routing-options static route 0.0.0.0/0
next-hop 172.16.0.13
set logical-systems P1 routing-instances sp2-route-table instance-type forwarding
set logical-systems P1 routing-instances sp2-route-table routing-options static route 0.0.0.0/0
next-hop 172.16.0.17
set logical-systems P1 routing-options rib-groups fbf-group import-rib inet.0
set logical-systems P1 routing-options rib-groups fbf-group import-rib sp1-route-table.inet.0
set logical-systems P1 routing-options rib-groups fbf-group import-rib sp2-route-table.inet.0
set logical-systems P2 interfaces lt-1/2/0 unit 2 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 2 peer-unit 1
set logical-systems P2 interfaces lt-1/2/0 unit 2 family inet address 172.16.0.2/30
set logical-systems P2 interfaces lt-1/2/0 unit 6 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 6 peer-unit 5
set logical-systems P2 interfaces lt-1/2/0 unit 6 family inet address 172.16.0.6/30
set logical-systems P2 interfaces lt-1/2/0 unit 9 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 9 peer-unit 10
set logical-systems P2 interfaces lt-1/2/0 unit 9 family inet address 172.16.0.9/30
set logical-systems P2 protocols ospf area 0.0.0.0 interface all
set logical-systems P2 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE1 interfaces lt-1/2/0 unit 14 encapsulation ethernet
set logical-systems PE1 interfaces lt-1/2/0 unit 14 peer-unit 13
set logical-systems PE1 interfaces lt-1/2/0 unit 14 family inet address 172.16.0.14/30
set logical-systems PE1 interfaces lo0 unit 3 family inet address 172.16.1.1/32
set logical-systems PE1 protocols ospf area 0.0.0.0 interface all
set logical-systems PE1 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE2 interfaces lt-1/2/0 unit 18 encapsulation ethernet
set logical-systems PE2 interfaces lt-1/2/0 unit 18 peer-unit 17
set logical-systems PE2 interfaces lt-1/2/0 unit 18 family inet address 172.16.0.18/30
set logical-systems PE2 interfaces lo0 unit 4 family inet address 172.16.2.2/32
set logical-systems PE2 protocols ospf area 0.0.0.0 interface all
set logical-systems PE2 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE3 interfaces lt-1/2/0 unit 1 encapsulation ethernet
set logical-systems PE3 interfaces lt-1/2/0 unit 1 peer-unit 2
set logical-systems PE3 interfaces lt-1/2/0 unit 1 family inet address 172.16.0.1/30
set logical-systems PE3 interfaces lo0 unit 1 family inet address 10.1.1.1/32
set logical-systems PE3 interfaces lo0 unit 1 family inet address 10.1.2.1/32
set logical-systems PE3 protocols ospf area 0.0.0.0 interface all
set logical-systems PE3 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE4 interfaces lt-1/2/0 unit 5 encapsulation ethernet
set logical-systems PE4 interfaces lt-1/2/0 unit 5 peer-unit 6
set logical-systems PE4 interfaces lt-1/2/0 unit 5 family inet address 172.16.0.5/30
```

```
set logical-systems PE4 interfaces lo0 unit 2 family inet address 10.2.1.1/32
set logical-systems PE4 interfaces lo0 unit 2 family inet address 10.2.2.1/32
set logical-systems PE4 protocols ospf area 0.0.0.0 interface all
set logical-systems PE4 protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Configuring the Firewall Filter on the Main Router

Step-by-Step Procedure

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

To configure the firewall filter on the main router:

1. Configure the source addresses for SP1 customers.

```
[edit firewall filter classify-customers term sp1-customers]
user@host# set from source-address 10.1.1.0/24
user@host# set from source-address 10.1.2.0/24
```

2. Configure the actions that are taken when packets are received with the specified source addresses.

   To track the action of the firewall filter, a log action is configured. The sp1-route-table.inet.0 routing table on Logical System P1 routes the packets.

```
[edit firewall filter classify-customers term sp1-customers]
user@host# set then log
user@host# set then logical-system P1 routing-instance sp1-route-table
```

3. Configure the source addresses for SP2 customers.

```
[edit firewall filter classify-customers term sp2-customers]
user@host# set from source-address 10.2.1.0/24
user@host# set from source-address 10.2.2.0/24
```

4. Configure the actions that are taken when packets are received with the specified source addresses.

To track the action of the firewall filter, a log action is configured. The sp2-route-table.inet.0 routing table on Logical System P1 routes the packet.

```
[edit firewall filter classify-customers term sp2-customers]
user@host# set then log
user@host# set then logical-system P1 routing-instance sp2-route-table
```

5. Configure the action to take when packets are received from any other source address.

   All of these packets are simply accepted and routed using the default IPv4 unicast routing table, inet.0.

```
[edit firewall filter classify-customers term default]
user@host# set then accept
```

Configuring the Routing Instances on the Logical System P1

Step-by-Step Procedure

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

To configure the routing instances on a logical system:

1. Configure the interfaces on the logical system.

```
[edit logical-systems P1 interfaces lt-1/2/0]
user@host# set unit 10 encapsulation ethernet
user@host# set unit 10 peer-unit 9
user@host# set unit 10 family inet address 172.16.0.10/30
user@host# set unit 13 encapsulation ethernet
user@host# set unit 13 peer-unit 14
user@host# set unit 13 family inet address 172.16.0.13/30
user@host# set unit 17 encapsulation ethernet
user@host# set unit 17 peer-unit 18
user@host# set unit 17 family inet address 172.16.0.17/30
```

2. Assign the `classify-customers` firewall filter to router interface lt-1/2/0.10 as an input packet filter.

```
[edit logical-systems P1 interfaces lt-1/2/0]
user@host# set unit 10 family inet filter input classify-customers
```

3. Configure connectivity, using either a routing protocol or static routing.

   As a best practice, disable routing on the management interface.

```
[edit  logical-systems P1 protocols ospf area 0.0.0.0]
user@host# set interface all
user@host# set interface fxp0.0 disable
```

4. Create the routing instances.

   These routing instances are referenced in the `classify-customers` firewall filter.

   The forwarding instance type provides support for filter-based forwarding, where interfaces are not associated with instances. All interfaces belong to the default instance, in this case Logical System P1.

```
[edit logical-systems P1 routing-instances]
user@host# set sp1-route-table instance-type forwarding
user@host# set sp2-route-table instance-type forwarding
```

5. Resolve the routes installed in the routing instances to directly connected next hops.

```
[edit logical-systems P1 routing-instances]
user@host# set sp1-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.13
user@host# set sp2-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.17
```

6. Group together the routing tables to form a routing table group.

   The first routing table, inet.0, is the primary routing table, and the additional routing tables are the secondary routing tables.

   The primary routing table determines the address family of the routing table group, in this case IPv4.

```
[edit logical-systems P1 routing-options]
user@host# set rib-groups fbf-group import-rib inet.0
```

```
user@host# set rib-groups fbf-group import-rib sp1-route-table.inet.0
user@host# set rib-groups fbf-group import-rib sp2-route-table.inet.0
```

7. Apply the routing table group to OSPF.

   This causes the OSPF routes to be installed into all the routing tables in the group.

```
[edit logical-systems P1 protocols ospf]
user@host# set rib-group fbf-group
```

8. If you are done configuring the device, commit the configuration.

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

Results

Confirm your configuration by issuing the `show firewall` and `show logical-systems P1` commands.

```
user@host# show firewall
filter classify-customers {
    term sp1-customers {
        from {
            source-address {
                10.1.1.0/24;
                10.1.2.0/24;
            }
        }
        then {
            log;
            logical-system P1 routing-instance sp1-route-table;
        }
    }
    term sp2-customers {
        from {
            source-address {
                10.2.1.0/24;
                10.2.2.0/24;
            }
        }
```

```
        then {
            log;
            logical-system P1 routing-instance sp2-route-table;
        }
    }
    term default {
        then accept;
    }
}
```

```
user@host# show logical-systems P1
interfaces {
    lt-1/2/0 {
        unit 10 {
            encapsulation ethernet;
            peer-unit 9;
            family inet {
                filter {
                    input classify-customers;
                }
                address 172.16.0.10/30;
            }
        }
        unit 13 {
            encapsulation ethernet;
            peer-unit 14;
            family inet {
                address 172.16.0.13/30;
            }
        }
        unit 17 {
            encapsulation ethernet;
            peer-unit 18;
            family inet {
                address 172.16.0.17/30;
            }
        }
    }
}
protocols {
    ospf {
```

```
        rib-group fbf-group;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {
    sp1-route-table {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 172.16.0.13;
            }
        }
    }
    sp2-route-table {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 172.16.0.17;
            }
        }
    }
}
routing-options {
    rib-groups {
        fbf-group {
            import-rib [ inet.0 sp1-route-table.inet.0 sp2-route-table.inet.0 ];
        }
    }
}
```

Verification

IN THIS SECTION

Pinging with Specified Source Addresses | **1248**

- Verifying the Firewall Filter | 1249

Confirm that the configuration is working properly.

Pinging with Specified Source Addresses

Purpose

Send some ICMP packets across the network to test the firewall filter.

Action

1. Log in to Logical System PE3.

```
user@host> set cli logical-system PE3
Logical system: PE3
```

2. Run the `ping` command, pinging the lo0.3 interface on Logical System PE1.

   The address configured on this interface is 172.16.1.1.

   Specify the source address 10.1.2.1, which is the address configured on the lo0.1 interface on Logical System PE3.

```
user@host:PE3> ping 172.16.1.1 source 10.1.2.1
PING 172.16.1.1 (172.16.1.1): 56 data bytes
64 bytes from 172.16.1.1: icmp_seq=0 ttl=62 time=1.444 ms
64 bytes from 172.16.1.1: icmp_seq=1 ttl=62 time=2.094 ms
^C
--- 172.16.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.444/1.769/2.094/0.325 ms
```

3. Log in to Logical System PE4.

```
user@host:PE3> set cli logical-system PE4
Logical system: PE4
```

4. Run the `ping` command, pinging the lo0.4 interface on Logical System PE2.

   The address configured on this interface is 172.16.2.2.

   Specify the source address 10.2.1.1, which is the address configured on the lo0.2 interface on Logical System PE4.

```
user@host:PE4> ping 172.16.2.2 source 10.2.1.1
PING 172.16.2.2 (172.16.2.2): 56 data bytes
64 bytes from 172.16.2.2: icmp_seq=0 ttl=62 time=1.473 ms
64 bytes from 172.16.2.2: icmp_seq=1 ttl=62 time=1.407 ms
^C
--- 172.16.2.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.407/1.440/1.473/0.033 ms
```

Meaning

Sending these pings activates the firewall filter actions.

Verifying the Firewall Filter

Purpose

Make sure the firewall filter actions take effect.

Action

1. Log in to Logical System P1.

```
user@host> set cli logical-system P1
Logical system: P1
```

2. Run the `show firewall log` command on Logical System P1.

```
user@host:P1> show firewall log
Log :
Time      Filter    Action Interface     Protocol        Src Addr           Dest Addr
13:52:20  pfe       A      lt-1/2/0.10   ICMP            10.2.1.1           172.16.2.2
13:52:19  pfe       A      lt-1/2/0.10   ICMP            10.2.1.1           172.16.2.2
13:51:53  pfe       A      lt-1/2/0.10   ICMP            10.1.2.1           172.16.1.1
13:51:52  pfe       A      lt-1/2/0.10   ICMP            10.1.2.1           172.16.1.1
```

RELATED DOCUMENTATION

- Configuring Filter-Based Forwarding
- Copying and Redirecting Traffic with Port Mirroring and Filter-Based Forwarding
- Example: Configuring Filter-Based Forwarding on the Source Address
- Using Filter-Based Forwarding to Export Monitored Traffic to Multiple Destinations
- Filter-Based Forwarding Overview

Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods

IN THIS SECTION

- Requirements | 1251
- Overview | 1251
- Configuration | 1252
- Verification | 1255

This example shows how to configure a stateless firewall filter that protects against ICMP denial-of-service attacks on a logical system.

Requirements

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

Overview

IN THIS SECTION

- Topology | **1251**

This example shows a stateless firewall filter called protect-RE that polices ICMP packets. The `icmp-policer` limits the traffic rate of the ICMP packets to 1,000,000 bps and the burst size to 15,000 bytes. Packets that exceed the traffic rate are discarded.

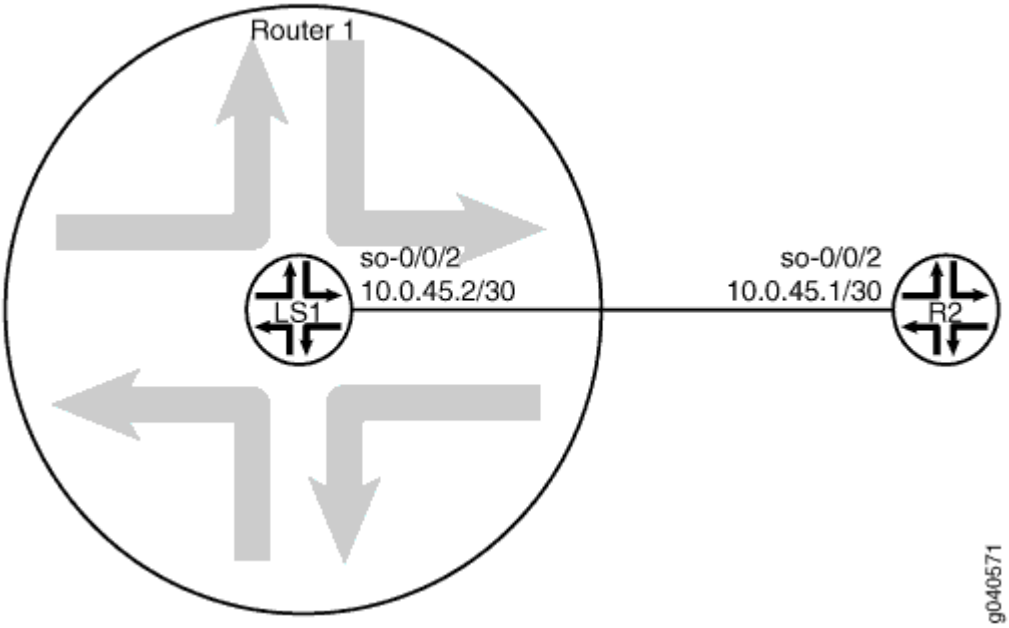
The policer is incorporated into the action of a filter term called `icmp-term`.

In this example, a ping is sent from a directly connected physical router to the interface configured on the logical system. The logical system accepts the ICMP packets if they are received at a rate of up to 1 Mbps (bandwidth-limit). The logical system drops all ICMP packets when this rate is exceeded. The `burst-size-limit` statement accepts traffic bursts up to 15 Kbps. If bursts exceed this limit, all packets are dropped. When the flow rate subsides, ICMP packets are again accepted.

Topology

Figure 56 on page 1252 shows the topology used in this example.

Figure 56: Logical System with a Stateless Firewall



Configuration

IN THIS SECTION

- CLI Quick Configuration | 1252
- Procedure | 1253
- Results | 1254

CLI Quick Configuration

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

```
set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet policer input icmp-policer
set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet address 10.0.45.2/30
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term from protocol icmp
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then policer icmp-
policer
```

```
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then accept
set logical-systems LS1 firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set logical-systems LS1 firewall policer icmp-policer if-exceeding burst-size-limit 15k
set logical-systems LS1 firewall policer icmp-policer then discard
```

Procedure

Step-by-Step Procedure

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

To configure an ICMP firewall filter on a logical system:

1. Configure the interface on the logical system.

```
[edit]
user@host# set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet address
10.0.45.2/30
```

2. Explicitly enable ICMP packets to be received on the interface.

```
[edit]
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term from
protocol icmp
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then
accept
```

3. Create the policer.

```
[edit]
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding bandwidth-limit
1m
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding burst-size-
limit 15k
user@host# set logical-systems LS1 firewall policer icmp-policer then discard
```

4. Apply the policer to a filter term.

```
[edit]
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then
policer icmp-policer
```

5. Apply the policer to the logical system interface.

```
[edit]
user@host# set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet policer input icmp-
policer
```

6. If you are done configuring the device, commit the configuration.

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

Results

Confirm your configuration by issuing the `show logical-systems LS1` command.

```
user@host# show logical-systems LS1
interfaces {
    so-0/0/2 {
        unit 0 {
            family inet {
                policer {
                    input icmp-policer;
                }
                address 10.0.45.2/30;
            }
        }
    }
}
firewall {
    family inet {
        filter protect-RE {
            term icmp-term {
                from {
```

```
                    protocol icmp;
                }
                then {
                    policer icmp-policer;
                    accept;
                }
            }
        }
    }
    policer icmp-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 15k;
        }
        then discard;
    }
}
```

Verification

IN THIS SECTION

- Verifying That Ping Works Unless the Limits Are Exceeded | **1255**

Confirm that the configuration is working properly.

Verifying That Ping Works Unless the Limits Are Exceeded

Purpose

Make sure that the logical system interface is protected against ICMP-based DoS attacks.

Action

Log in to a system that has connectivity to the logical system and run the `ping` command.

```
user@R2> ping 10.0.45.2
PING 10.0.45.2 (10.0.45.2): 56 data bytes
```

```
64 bytes from 10.0.45.2: icmp_seq=0 ttl=64 time=1.316 ms
64 bytes from 10.0.45.2: icmp_seq=1 ttl=64 time=1.277 ms
64 bytes from 10.0.45.2: icmp_seq=2 ttl=64 time=1.269 ms
```

```
user@R2> ping 10.0.45.2 size 20000
PING 10.0.45.2 (10.0.45.2): 20000 data bytes
^C
--- 10.0.45.2 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Meaning

When you send a normal ping, the packet is accepted. When you send a ping packet that exceeds the filter limit, the packet is discarded.

RELATED DOCUMENTATION

- Example: Creating an Interface on a Logical System

Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods

IN THIS SECTION

- Requirements | 1257
- Overview | 1257
- Configuration | 1258
- Verification | 1261

This example shows how to configure a stateless firewall filter that protects against ICMP denial-of-service attacks on a logical system.

Requirements

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

Overview

IN THIS SECTION

- Topology | **1257**

This example shows a stateless firewall filter called protect-RE that polices ICMP packets. The **icmp-policer** limits the traffic rate of the ICMP packets to 1,000,000 bps and the burst size to 15,000 bytes. Packets that exceed the traffic rate are discarded.

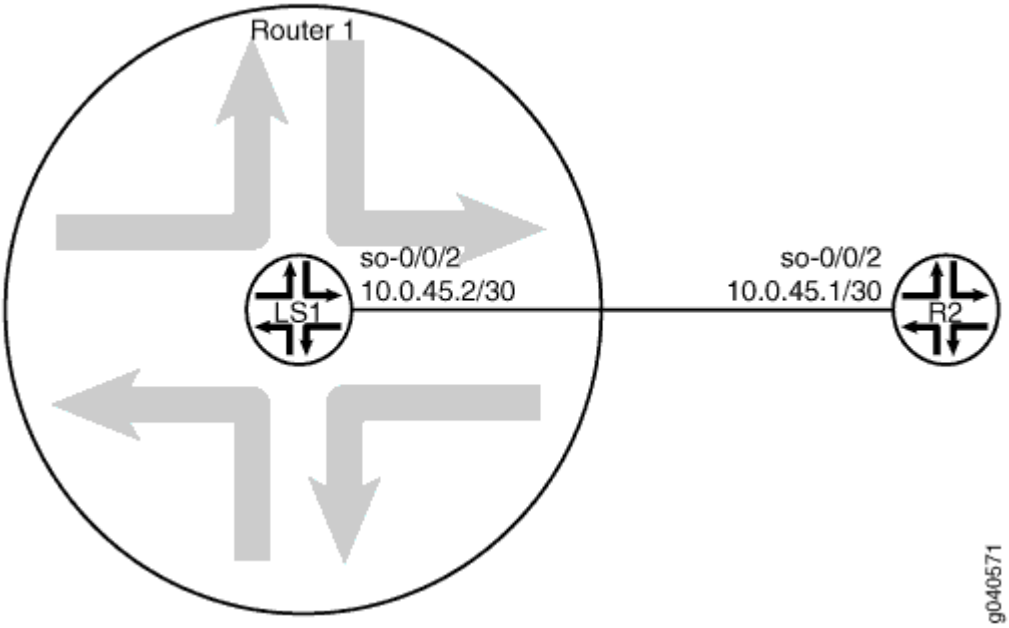
The policer is incorporated into the action of a filter term called **icmp-term**.

In this example, a ping is sent from a directly connected physical router to the interface configured on the logical system. The logical system accepts the ICMP packets if they are received at a rate of up to 1 Mbps (bandwidth-limit). The logical system drops all ICMP packets when this rate is exceeded. The `burst-size-limit` statement accepts traffic bursts up to 15 Kbps. If bursts exceed this limit, all packets are dropped. When the flow rate subsides, ICMP packets are again accepted.

Topology

Figure 57 on page 1258 shows the topology used in this example.

Figure 57: Logical System with a Stateless Firewall



Configuration

IN THIS SECTION

- CLI Quick Configuration | **1258**
- Procedure | **1259**
- Results | **1260**

CLI Quick Configuration

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

```
set logical-systems LS1 interfaces ge-0/0/2 unit 0 family inet policer input icmp-policer
set logical-systems LS1 interfaces ge-0/0/2 unit 0 family inet address 10.0.45.2/30
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term from protocol icmp
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then policer icmp-
policer
```

```
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then accept
set logical-systems LS1 firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set logical-systems LS1 firewall policer icmp-policer if-exceeding burst-size-limit 15k
set logical-systems LS1 firewall policer icmp-policer then discard
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure an ICMP firewall filter on a logical system:

1. Configure the interface on the logical system.

```
[edit]
user@host# set logical-systems LS1 interfaces ge-0/0/2 unit 0 family inet address
10.0.45.2/30
```

2. Explicitly enable ICMP packets to be received on the interface.

```
[edit]
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term from
protocol icmp
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then
accept
```

3. Create the policer.

```
[edit]
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding bandwidth-limit
1m
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding burst-size-
limit 15k
user@host# set logical-systems LS1 firewall policer icmp-policer then discard
```

4. Apply the policer to a filter term.

```
[edit]
user@host# set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then
policer icmp-policer
```

5. Apply the policer to the logical system interface.

```
[edit]
user@host# set logical-systems LS1 interfaces ge-0/0/2 unit 0 family inet policer input icmp-
policer
```

6. If you are done configuring the device, commit the configuration.

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

Results

Confirm your configuration by issuing the `show logical-systems LS1` command.

```
user@host# show logical-systems LS1
interfaces {
    ge-0/0/2 {
        unit 0 {
            family inet {
                policer {
                    input icmp-policer;
                }
                address 10.0.45.2/30;
            }
        }
    }
}
firewall {
    family inet {
        filter protect-RE {
            term icmp-term {
                from {
```

```
                    protocol icmp;
                }
                then {
                    policer icmp-policer;
                    accept;
                }
            }
        }
    }
    policer icmp-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 15k;
        }
        then discard;
    }
}
```

Verification

IN THIS SECTION

- Verifying That Ping Works Unless the Limits Are Exceeded | **1261**

Confirm that the configuration is working properly.

Verifying That Ping Works Unless the Limits Are Exceeded

Purpose

Make sure that the logical system interface is protected against ICMP-based DoS attacks.

Action

Log in to a system that has connectivity to the logical system and run the `ping` command.

```
user@R2> ping 10.0.45.2
PING 10.0.45.2 (10.0.45.2): 56 data bytes
```

```
64 bytes from 10.0.45.2: icmp_seq=0 ttl=64 time=1.316 ms
64 bytes from 10.0.45.2: icmp_seq=1 ttl=64 time=1.277 ms
64 bytes from 10.0.45.2: icmp_seq=2 ttl=64 time=1.269 ms
```

```
user@R2> ping 10.0.45.2 size 20000
PING 10.0.45.2 (10.0.45.2): 20000 data bytes
^C
--- 10.0.45.2 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Meaning

When you send a normal ping, the packet is accepted. When you send a ping packet that exceeds the filter limit, the packet is discarded.

RELATED DOCUMENTATION

- Example: Creating an Interface on a Logical System

Unsupported Firewall Filter Statements for Logical Systems

Table 63 on page 1263 shows statements that are supported at the `[edit firewall]` hierarchy level but not at the `[edit logical-systems` *`logical-system-name`* `firewall]` hierarchy level.

Table 63: Unsupported Firewall Statements for Logical Systems

Statement	Example	Description

accounting-profile	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter myfilter {
	accounting-profile fw-profile;
	...
	term accept-all {
	then {
	count counter1;
	accept;
	}
	}
	}
	}
	}
	}
	}
		In this example, the accounting-profile statement is not allowed because the accounting profile fw-profile is configured under the [edit accounting-options] hierarchy.

hierarchical-policer	[edit]
	logical-systems {
	lr1 {
	firewall {
	hierarchical-policer {
	...
	}
	}
	}
	}
		In this example, the hierarchical policer statement requires a class-of-service configuration, which is not supported under logical systems.

Table 63: Unsupported Firewall Statements for Logical Systems *(Continued)*

Statement	Example	Description

load-balance-group	[edit]
	logical-systems {
	    ls1 {
	        firewall {
	            load-balance-group lb-group {
	                next-hop-group nh-group;
	            }
	        }
	    }
	}
		This configuration is not allowed because the next-hop-group nh-group statement must be configured at the [edit forwarding-options next-hop-group] hierarchy level—outside the [edit logical-systems *logical-system-name* firewall] hierarchy.
		
		Currently, the forwarding-options dhcp-relay statement is the only forwarding option supported for logical systems.

virtual-channel	[edit]
	logical-systems {
	    ls1 {
	        firewall {
	            family inet {
	                filter foo {
	                    term one {
	                        from {
	                            source-address
	10.1.0.0/16;
	                        }
	                        then {
	                            virtual-channel
	sammy;
	                        }
	                    }
	                }
	            }
	        }
	    }
	}
		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.
		
		NOTE:
		The virtual-channel statement is supported for J Series devices only, provided the firewall filter is configured outside of a logical-system.

RELATED DOCUMENTATION

Firewall Filters in Logical Systems Overview | 1213
Guidelines for Configuring and Applying Firewall Filters in Logical Systems | 1215
Unsupported Actions for Firewall Filters in Logical Systems | 1265
Introduction to Logical Systems
Junos OS Logical Systems User Guide for Routing Devices
Logical Systems Operations and Restrictions
Junos OS Logical Systems User Guide for Routing Devices

Unsupported Actions for Firewall Filters in Logical Systems

Table 64 on page 1265 describes the firewall filter actions that are supported at the `[edit firewall]` hierarchy level, but not supported at the `[edit logical-systems` *`logical-system-name`* `firewall]` hierarchy level.

Table 64: Unsupported Actions for Firewall Filters in Logical Systems

Firewall Filter Action	Example	Description
Terminating Actions Not Supported in a Logical System		

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

`logical-system`	`[edit]`
	`logical-systems {`
	`  ls1 {`
	`  firewall {`
	`  family inet {`
	`  filter foo {`
	`  term one {`
	`  from {`
	`  source-address 10.1.0.0/16;`
	`  }`
	`  then {`
	`  logical-system fred;`
	`  }`
	`  }`
	`  }`
	`  }`
	`  }`
	`  }`
	`}`
		Because the `logical-system` action refers to `fred`—a logical system defined outside the local logical system—, this action is not supported.

Nonterminating Actions Not Supported in a Logical System		

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

ipsec-sa	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter foo {
	term one {
	from {
	source-address
	10.1.0.0/16;
	}
	then {
	ipsec-sa barney;
	}
	}
	}
	}
	}
	}
	}
		Because the ipsec-sa action modifier references barney—a security association defined outside the local logical system—this action is not supported.

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

next-hop-group	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter foo {
	term one {
	from {
	source-address
	10.1.0.0/16;
	}
	then {
	next-hop-group
	fred;
	}
	}
	}
	}
	}
	}
	}
		Because the next-hop-group action refers to fred—an object defined at the [edit forwarding-options next-hop-group] hierarchy level—this action is not supported.

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

port-mirror	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter foo {
	term one {
	from {
	source-address
	10.1.0.0/16;
	}
	then {
	port-mirror;
	}
	}
	}
	}
	}
	}
	}
		Because the port-mirror action relies on a configuration defined at the [edit forwarding-options port-mirroring] hierarchy level, this action is not supported.

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

sample	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter foo {
	term one {
	from {
	source-address
	10.1.0.0/16;
	}
	then {
	sample;
	}
	}
	}
	}
	}
	}
	}
		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.

Table 64: Unsupported Actions for Firewall Filters in Logical Systems *(Continued)*

Firewall Filter Action	Example	Description

syslog	[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;
	}
	}
	}
	}
	}
	}
		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.
		
		Because this firewall configuration relies on a configuration outside the logical system, the `syslog` action modifier is not supported.

RELATED DOCUMENTATION

- Firewall Filters in Logical Systems Overview | 1213
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems | 1215
- Unsupported Firewall Filter Statements for Logical Systems | 1262
- Introduction to Logical Systems
- Logical Systems Operations and Restrictions

Filter-Based Forwarding for Routing Instances

You can use stateless firewall filters in routing instances to control how packets travel in a network for IPv4 and IPv6 traffic. This is called filter-based forwarding.

You can define a firewall filtering term that directs matching packets to a specified routing instance. This type of filtering can be configured to route specific types of traffic through a firewall or other security device before the traffic continues on its path. To configure a stateless *firewall filter* to direct traffic to a routing instance, configure a term with the `routing-instance` *routing-instance-name* terminating action at the `[edit firewall family <inet | inet6>]` hierarchy level to specify the routing instance to which matching packets will be forwarded. You can apply a forwarding table filter to a routing instance of type forwarding and also to the default routing instance `inet.0`. To configure the filter to direct traffic to the master routing instance, use the `routing-instance default` statement at the `[edit firewall family <inet | inet6>]` hierarchy level.

The following limitations apply to filter-based forwarding table configured on routing instances:

- You cannot configure any of the following actions in a firewall filtering term when the filtering term contains the `routing-instance` *routing-instance-name* terminating action:
  - `count` *counter-name*
  - `discard`
  - `forwarding-class` *class-name*
  - `log`
  - `loss-priority (high | medium-high | low)`
  - `policer` *policer-name*
  - `port-mirror`
  - `reject` *message-type*
  - `syslog`
  - `three-color-policer (single-rate | two-rate)` *policer-name*
- You cannot configure the `fragment-flags` *number* match condition in the filter term.
- You cannot attach a filter that is either default or physical interface-specific.
- You cannot attach a filter to the egress direction of routing instances.
- IPv6 filter-based forwarding does not support the following L4 matches:

- source-port
- destination-port
- icmp-type
- icmp-code

Although you can configure forwarding of packets from one VRF to another VRF, you cannot configure forwarding from a VRF to the global routing instance.

The maximum number of routing instances supported is 64, which is the same as the maximum number of virtual routers supported. Forwarding packets to the global table (default VRF) is not supported for filter-based forwarding.

NOTE: Filter-based forwarding on the interface will not work when source MAC address filter is configured because the source MAC address filter takes higher precedence over filter-based forwarding.

RELATED DOCUMENTATION

Example: Configuring Filter-Based Forwarding on the Source Address | 1497

Forwarding Table Filters for Routing Instances on ACX Series Routers

Forwarding table filter is a mechanism by which all the packets forwarded by a certain forwarding table are subjected to filtering and if a packet matches the filter condition, the configured action is applied on the packet. You can use the forwarding table filter mechanism to apply a filter on all interfaces associated with a single routing instance with a simple configuration. You can apply a forwarding table filter to a routing instance of type forwarding and also to the default routing instance `inet.0`. To configure a forwarding table filter, include the `filter` *`filter-name`* statement at the `[edit firewall family <inet | inet6>]` hierarchy level.

The following limitations apply to forwarding table filters configured on routing instances:

- You cannot attach the same filter to more than one routing instance.
- You cannot attach the same filter at both the `[edit interfaces` *`interface-name`* `family <inet | inet6> filter input` *`filter-name`*`]` and `[edit routing-instances` *`instance-name`* `forwarding-options family <inet | inet6> filter input` *`filter-name`*`]` hierarchy level.

- You cannot attach a filter that is either interface-specific or a physical interface filter.
- You cannot attach a filter to the egress direction of routing instances.

RELATED DOCUMENTATION

Configuring Forwarding Table Filters | 1274

Configuring 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.

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter; it essentially controls which bearer packets the router accepts and forwards. To configure a forwarding table filter, include the `firewall` statement at the `[edit]` hierarchy level:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            term term-name {
```

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

family-name is the family address type: IPv4 (**inet**), IPv6 (**inet6**), Layer 2 traffic (**bridge**), or MPLS (**mpls**).

term-name is a named structure in which match conditions and actions are defined.

match-conditions are the criteria against which a bearer packet is compared; for example, the IP address of a source device or a destination device. You can specify multiple criteria in a match condition.

action specifies what happens if a packet matches all criteria; for example, the gateway GPRS support node (GGSN) accepting the bearer packet, performing a lookup in the forwarding table, and forwarding the packet to its destination; discarding the packet; and discarding the packet and returning a rejection message.

action-modifiers are actions that are taken in addition to the GGSN accepting or discarding a packet when all criteria match; for example, counting the packets and logging a packet.

To create a forwarding table, include the `instance-type` statement with the **forwarding** option at the `[edit routing-instances` *`instance-name`*`]` hierarchy level:

```
[edit]
routing-instances instance-name {
    instance-type forwarding;
}
```

To apply a forwarding table filter to a VPN routing and forwarding (VRF) table, include the **filter** and `input` statements at the `[edit routing-instance` *`instance-name`* `forwarding-options family` *`family-name`*`]` hierarchy level:

```
[edit routing-instances instance-name]
instance-type forwarding;
forwarding-options {
    family family-name {
        filter {
```

```
            input filter-name;
        }
    }
}
```

To apply a forwarding table filter to a forwarding table, include the **filter** and `input` statements at the `[edit forwarding-options family family-name]` hierarchy level:

```
[edit forwarding-options family family-name]
filter {
    input filter-name;
}
```

To apply a forwarding table filter to the default forwarding table **inet.0**, which is not associated with a specific routing instance, include the **filter** and `input` statements at the `[edit forwarding-options family inet]` hierarchy level:

```
[edit forwarding-options family inet]
filter {
    input filter-name;
}
```

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters | **814**

Guidelines for Applying Standard Firewall Filters | **821**

Applying Forwarding Table Filters

CHAPTER 18

Configuring Firewall Filter Accounting and Logging

IN THIS CHAPTER

- Accounting for Firewall Filters Overview | 1277
- System Logging Overview | 1278
- System Logging of Events Generated for the Firewall Facility | 1279
- Firewall Filter Logging Actions | 1282
- Example: Configuring Statistics Collection for a Firewall Filter | 1286
- Example: Configuring Logging for a Firewall Filter Term | 1293

Accounting for Firewall Filters Overview

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*, source class and destination class usage, and Routing Engine. If you apply the same profile name to both a firewall filter and an interface, it causes an error.

RELATED DOCUMENTATION

Example: Configuring Statistics Collection for a Firewall Filter | 1286

System Logging Overview

The Junos OS generates system log messages (also called *syslog messages*) to record *system events* that occur on the device. Events consist of routine operations, failure and error conditions, and critical conditions that might require urgent resolution. This system logging utility is similar to the UNIX `syslogd` utility.

Each Junos OS system log message belongs to a message category, called a *facility*, that reflects the hardware- or software-based source of the triggering event. A group of messages belonging to the same facility are either generated by the same software process or concern a similar hardware condition or user activity (such as authentication attempts). Each system log message is also preassigned a *severity*, which indicates how seriously the triggering event affects router (or switch) functions. Together, the facility and severity of an event are known as the message *priority*. The content of a syslog message identifies the Junos OS *process* that generates the message and briefly describes the operation or error that occurred.

By default, syslog messages that have a severity of `info` or more serious are written to the main system log file `messages` in the `/var/log` directory of the local Routing Engine. To configure global settings and facility-specific settings that override these default values, you can include statements at the `[edit system syslog]` hierarchy level.

For all syslog facilities or for a specified facility, you can configure the syslog message utility to redirect messages of a specified severity to a specified file instead of to the main system log file. You can also configure the syslog message utility to write syslog messages of a specified severity, for all syslog facilities or for a specified facility, to additional destinations. In addition to writing syslog messages to a log file, you can write syslog messages to the terminal sessions of any logged-in users, to the router (or switch) console, or to a remote host or the other Routing Engine.

At the global level—for all system logging messages, regardless of facility, severity, or destination—you can override the default values for file-archiving properties and the default timestamp format.

RELATED DOCUMENTATION

System Logging of Events Generated for the Firewall Facility | **1279**

Firewall Filter Logging Actions | **1282**

Example: Configuring Logging for a Firewall Filter Term | **1293**

System Logging of Events Generated for the Firewall Facility

System log messages generated for *firewall filter* actions belong to the `firewall` facility. Just as you can for any other Junos OS system logging facility, you can direct `firewall` facility syslog messages to one or more specific destinations: to a specified file, to the terminal session of one or more logged in users (or to all users), to the router (or switch) console, or to a remote host or the other Routing Engine on the router (or switch).

When you configure a syslog message destination for `firewall` facility syslog messages, you include a statement at the `[edit system syslog]` hierarchy level, and you specify the `firewall` facility name together with a severity level. Messages from the `firewall` that are rated at the specified level or more severe are logged to the destination.

System log messages with the `DFWD_` prefix are generated by the firewall process (`dfwd`), which manages compilation and downloading of Junos OS firewall filters. System log messages with the `PFE_FW_` prefix are messages about firewall filters, generated by the Packet Forwarding Engine controller, which manages packet forwarding functions. For more information, see the System Log Explorer.

Table 65 on page 1280 lists the system log destinations you can configure for the `firewall` facility.

Table 65: Syslog Message Destinations for the Firewall Facility

Destination	Description	Configuration Statements Under [edit system syslog]

File	Configuring this option keeps the `firewall` syslog messages out of the main system log file.
	
	To include priority and facility with messages written to the file, include the `explicit-priority` statement.
	
	To override the default standard message format, which is based on a UNIX system log format, include the `structured-data` statement. When the `structured-data` statement is included, other statements that specify the format for messages written to the file are ignored (the `explicit-priority` statement at the `[edit system syslog file filename]` hierarchy level and the `time-format` statement at the `[edit system syslog]` hierarchy level).
		file *filename* {
		  firewall *severity*;
		  allow-duplicates;
		  archive *archive-options*;
		  explicit-priority;
		  structured-data;
		}
		allow-duplicates;
		archive *archive-options*;
		time-format (*option*);

Terminal session	Configuring this option causes a copy of the `firewall` syslog messages to be written to the specified terminal sessions. Specify one or more user names, or specify `*` for all logged in users.	**user** (*username* | *****) {
		  firewall *severity*;
		}
		time-format (*option*);

Router (or switch) console	Configuring this option causes a copy of the **firewall** syslog messages to be written to the router (or switch) console.	**console** {
		  firewall *severity*;
		}
		time-format (*option*);

Table 65: Syslog Message Destinations for the Firewall Facility *(Continued)*

Destination	Description	Configuration Statements Under [edit system syslog]

Remote host or the other Routing Engine	Configuring this option causes a copy of the `firewall` syslog messages to be written to the specified remote host or to the other Routing Engine.
	
	To override the default alternative facility for forwarding `firewall` syslog messages to a remote machine (`local3`), include the `facility-override firewall` statement.
	
	To include priority and facility with messages written to the file, include the `explicit-priority` statement.
		host (*hostname* | **other-routing-engine**) {
		  firewall ***severity***;
		  allow-duplicates;
		  archive ***archive-options***;
		  facility-override firewall;
		  explicit-priority;
		}
		allow-duplicates; # All destinations
		archive ***archive-options***;
		time-format (***option***);

By default, the timestamp recorded in a standard-format system log message specifies the month, date, hour, minute, and second when the message was logged, as in the example:

```
Sep 07 08:00:10
```

To include the year, the millisecond, or both in the timestamp for all system logging messages, regardless of the facility, include one of the following statement at the `[edit system syslog]` hierarchy level:

- `time-format year;`
- `time-format millisecond;`
- `time-format year millisecond;`

The following example illustrates the format for a timestamp that includes both the millisecond (`401`) and the year (`2010`):

```
Sep 07 08:00:10.401.2010
```

RELATED DOCUMENTATION

System Logging Overview | 1278

Firewall Filter Logging Actions | 1282

Example: Configuring Logging for a Firewall Filter Term | 1293

Junos OS System Logging Facilities and Message Severity Levels

Junos OS System Log Configuration Hierarchy

Junos OS Default System Log Settings

Logging Messages in Structured-Data Format

Including the Year or Millisecond in Timestamps

Changing the Alternative Facility Name for System Log Messages Directed to a Remote Destination

Alternate Facilities for System Log Messages Directed to a Remote Destination

Firewall Filter Logging Actions

For IPv4 and IPv6 firewall filters, you can configure the filter to write a summary of matching packet headers to the log or syslog by specifying either the **syslog** or **log** action. The main difference between the two is the permanence of the record. Logs are only buffered in memory, and when that buffer is full, the oldest records are replaced with new ones as they come in. Syslogs, on the other hand, can be saved to disk or forwarded to a remote syslog server. In both cases, a summary of the packet header is logged (not a copy of the packet itself). Service filters and simple filters do not support either the **log** or **syslog** action.

NOTE: Both the **syslog** and **log** actions can consume significant CPU and/or disk space on the device. Juniper recommends that you off-load logs by writing them to a remote syslog server, and that you constrain logging by using it for diagnostics only.

Syslog

As noted, system logs can be written to disk and/or sent to a remote server. Saved logs are written to the `/var/log` directory. You can view a list of all available log files on the device by running the `show log` command without options. Note, that within a given log file, the firewall action logs may be interspersed with event messages.

The following **syslog** configuration shows system logs being sent to a remote server at 172.27.1.1, and also save them to a file named “firewall” on the local device.

```
host@device-RE0# show system syslog
host 172.27.1.1 {
    firewall any;
}
<...>
file firewall {
    firewall any;
}
```

To view system logs, run the `show syslog message` command.

To view the contents of a given system log file, run either the `show log` *`filename`* or the `file show /var/log/` *`filename`* command.

To clear system log file contents, run the `clear log` *`filename`* command. You can include the `all` option to delete all saved logs, including records being written to the current log file.

Configuration details are shown here:

```
firewall {
    family {
        filter filter-name {
            from {
                match-conditions;
            }
            then {
                ...
                syslog;
                terminating-action;
            }
        }
    }
}
```

Log

The **log** action writes log information to a buffer. There is no option for writing logs to a remote server, or for writing them to disk. Once the available buffer is full, new logs will replace the oldest, so a historical record is not kept. Logs are cleared whenever the device or PFE is restarted.

Configuration details are shown here:

```
firewall {
    family {
        filter filter-name {
            from {
                match-conditions;
            }
            then {
                ...
                log;
                terminating-action;
            }
        }
    }
}
```

To view the logs, run the `show firewall log` command.

Log Details

The following shows what kind of information is typically included in syslog and log entries:

```
user@host> show log messages_firewall_any

Mar 20 08:08:45 hostname feb FW: ge-1/1/0.0 A icmp 192.168.207.222 192.168.207.223  0  0 (1
packets)
```

The fields are explained here:

- `Date and Time`—Date and time at which the packet was received (not shown in the default).

  `Hostname`—Name of the device on which the match occurred..

  `Interface`—Physical interface that the packet traversed.

- Filter action. In the example above, it is A.
  - `A`—Accept (or next term)
  - `D`—Discard
  - `R`—Reject

- `Protocol`—Packet protocol. May be a name or number, and may also include the source and destination ports. In the example above, the protocol is ICMP, which may then include the ICMP type and code.
- `Source address`—Source IP address of the packet.
- `Destination address`—Destination IP address of the packet.
- `Source port`—Source port of the packet (TCP and UDP packets only). In the example above, the port is 0.
- `Destination port`—Destination port of the packet (TCP and UDP packets only). In the example above, the port is 0.
- `Packets in sample interval`—This example show only one matching packet was detected in the sample interval (about a second). If packets arrive at faster rate, the system log automatically compresses the information so that less output is generated.

RELATED DOCUMENTATION

System Logging Overview | **1278**

System Logging of Events Generated for the Firewall Facility | **1279**

Example: Configuring Logging for a Firewall Filter Term | **1293**

System Log Explorer

System Log Explorer

Example: Configuring Statistics Collection for a Firewall Filter

IN THIS SECTION

- Requirements | 1286
- Overview | 1286
- Configuration | 1287
- Verification | 1293

This example shows how to configure and apply a firewall filter that collects data according to parameters specified in an associated accounting profile.

Requirements

Firewall filter accounting profiles are supported for all traffic types except `family any`.

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1286

In this example, you create a firewall filter accounting profile and apply it to a firewall filter. The accounting profile specifies how frequently to collect packet and byte count statistics and the name of the file to which the statistics are written. The profile also specifies that statistics are to be collected for three firewall filter counters.

Topology

The firewall filter accounting profile `filter_acctg_profile` specifies that statistics are collected every 60 minutes, and the statistics are written to the file `/var/log/ff_accounting_file`. Statistics are collected for counters named `counter1`, `counter2`, and `counter3`.

The IPv4 firewall filter named `my_firewall_filter` increments a counter for each of three filter terms. The filter is applied to logical interface `ge-0/0/1.0`.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1287**
- Configure an Accounting Profile | **1288**
- Configure a Firewall Filter That References the Accounting Profile | **1289**
- Apply the Firewall Filter to an Interface | **1290**
- Confirm Your Candidate Configuration | **1290**
- Clear the Counters and Commit Your Candidate Configuration | **1292**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set accounting-options file ff_accounting_file files 10
set accounting-options file ff_accounting_file transfer-interval 10
set accounting-options filter-profile filter_acctg_profile file ff_accounting_file
set accounting-options filter-profile filter_acctg_profile interval 60
set accounting-options filter-profile filter_acctg_profile counters counter1
set accounting-options filter-profile filter_acctg_profile counters counter2
set accounting-options filter-profile filter_acctg_profile counters counter3
set firewall family inet filter my_firewall_filter accounting-profile filter_acctg_profile
set firewall family inet filter my_firewall_filter term term1 from protocol ospf
set firewall family inet filter my_firewall_filter term term1 then count counter1
set firewall family inet filter my_firewall_filter term term1 then discard
set firewall family inet filter my_firewall_filter term term2 from source-address 10.108.0.0/16
set firewall family inet filter my_firewall_filter term term2 then count counter2
set firewall family inet filter my_firewall_filter term term2 then discard
```

```
set firewall family inet filter my_firewall_filter term accept-all then count counter3
set firewall family inet filter my_firewall_filter term accept-all then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input my_firewall_filter
```

Configure an Accounting Profile

Step-by-Step Procedure

To configure an accounting profile:

1. Create a log file to associate with the accounting profile.

```
[edit]
user@host# edit accounting-options file ff_accounting_file files 10
edit accounting-options file ff_accounting_file transfer-interval 10
```

2. Create the accounting profile `filter_acctg_profile`.

```
[edit]
user@host# edit accounting-options filter-profile filter_acctg_profile
```

3. Configure the accounting profile to filter and collect packet and byte count statistics every 60 minutes and write them to the `/var/log/ff_accounting_file` file.

```
[edit accounting-options filter-profile filter_acctg_profile]
user@host# set file ff_accounting_file
user@host# set interval 60
```

4. Configure the accounting profile to collect filter profile statistics (packet and byte counts) for three counters.

```
[edit accounting-options filter-profile filter_acctg_profile]
user@host# set counters counter1
user@host# set counters counter2
user@host# set counters counter3
```

Configure a Firewall Filter That References the Accounting Profile

Step-by-Step Procedure

To configure a firewall filter that references the accounting profile:

1. Create the firewall filter `my_firewall_filter`.

```
[edit]
user@host# edit firewall family inet filter my_firewall_filter
```

2. Apply the filter-accounting profile `filter_acctg_profile` to the firewall filter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set accounting-profile filter_acctg_profile
```

3. Configure the first filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term term1 from protocol ospf
user@host# set term term1 then count counter1
user@host# set term term1 then discard
```

4. Configure the second filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term term2 from source-address 10.108.0.0/16
user@host# set term term2 then count counter2
user@host# set term term2 then discard
```

5. Configure the third filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term accept-all then count counter3
user@host# set term accept-all then accept
```

Apply the Firewall Filter to an Interface

Step-by-Step Procedure

To apply the firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input my_firewall_filter
```

Confirm Your Candidate Configuration

Step-by-Step Procedure

To confirm your candidate configuration:

1. Confirm the configuration of the accounting profile by entering the `show accounting-options` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show accounting-options
file ff_accounting_file {
    files 10;
    transfer-interval 10;
}
filter-profile filter_acctg_profile {
    file ff_accounting_file;
```

```
    interval 60;
    counters {
        counter1;
        counter2;
        counter3;
    }
}
```

2. Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter my_firewall_filter {
        accounting-profile filter_acctg_profile;
        term term1 {
            from {
                protocol ospf;
            }
            then {
                count counter1;
                discard;
            }
        }
        term term2 {
            from {
                source-address {
                    10.108.0.0/16;
                }
            }
            then {
                count counter2;
                discard;
            }
        }
        term accept-all {
            then {
                count counter3;
                accept;
            }
```

```
        }
    }
}
```

3. Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input my_firewall_filter;
            }
            address 10.1.2.3/30;
        }
    }
}
```

Clear the Counters and Commit Your Candidate Configuration

Step-by-Step Procedure

To clear the counters and commit your candidate configuration:

1. From operational command mode, use the `clear firewall all` command to clear the statistics for all firewall filters.

   To clear only the counters incremented in this example, include the name of the firewall filter.

```
[edit]
user@host> clear firewall filter my_firewall_filter
```

2. Commit your candidate configuration.

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

Verification

To verify that the filter is applied to the logical interface, run the `show interfaces` command with the `detail` or `extensive` output level.

To verify that the three counters are collected separately, run the `show firewall filter my_firewall_filter` command.

```
user@host> show firewall filter my_firewall_filter

Filter: my_firewall_filter
Counters:
Name                                                Bytes              Packets
counter1                                                0                    0
counter2                                                0                    0
counter3                                                0                    0
```

RELATED DOCUMENTATION

Accounting for Firewall Filters Overview | 1277

Example: Configuring Logging for a Firewall Filter Term

IN THIS SECTION

- Requirements | 1294
- Overview | 1294
- Configuration | 1294
- Verification | 1298

This example shows how to configure a firewall filter to log packet headers.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a firewall filter that logs and counts ICMP packets that have `192.168.207.222` as either their source or destination.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1294**
- Configure the Syslog Messages File for the Firewall Facility | **1295**
- Configure the Firewall Filter | **1295**
- Apply the Firewall Filter to a Logical Interface | **1296**
- Confirm and Commit Your Candidate Configuration | **1297**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set system syslog file messages_firewall_any firewall any
set system syslog file messages_firewall_any archive no-world-readable
set firewall family inet filter icmp_syslog term icmp_match from address 192.168.207.222/32
set firewall family inet filter icmp_syslog term icmp_match from protocol icmp
set firewall family inet filter icmp_syslog term icmp_match then count packets
set firewall family inet filter icmp_syslog term icmp_match then syslog
set firewall family inet filter icmp_syslog term icmp_match then accept
set firewall family inet filter icmp_syslog term default_term then accept
```

```
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input icmp_syslog
```

Configure the Syslog Messages File for the Firewall Facility

Step-by-Step Procedure

To configure a syslog messages file for the `firewall` facility:

1. Configure a messages file for all syslog messages generated for the `firewall` facility.

```
user@host# set system syslog file messages_firewall_any firewall any
```

2. Restrict permission to the archived `firewall` facility syslog files to the root user and users who have the Junos OS maintenance permission.

```
user@host# set system syslog file messages_firewall_any archive no-world-readable
```

Configure the Firewall Filter

Step-by-Step Procedure

To configure the firewall filter `icmp_syslog` that logs and counts ICMP packets that have `192.168.207.222` as either their source or destination:

1. Create the firewall filter `icmp_syslog`.

```
[edit]
user@host# edit firewall family inet filter icmp_syslog
```

2. Configure matching on the ICMP protocol and an address.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term icmp_match from address 192.168.207.222/32
user@host# set term icmp_match from protocol icmp
```

3. Count, log,, and accept matching packets.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term icmp_match then count packets
user@host# set term icmp_match then syslog
user@host# set term icmp_match then accept
```

4. Accept all other packets.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term default_term then accept
```

Apply the Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input icmp_syslog
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the syslog message file for the `firewall` facility by entering the `show system` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show system
syslog {
    file messages_firewall_any {
        firewall any;
        archive no-world-readable;
    }
}
```

2. Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show 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_term {
```

```
            then accept;
        }
    }
}
```

3. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input icmp_syslog;
            }
            address 10.1.2.3/30;
        }
    }
}
```

4. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show log filter` command:

```
user@host> show log messages_firewall_any
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 1-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:

```
user@host> show log messages_firewall_any
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)
```

RELATED DOCUMENTATION

System Logging Overview | 1278

Firewall Filter Logging Actions | 1282

System Log Explorer

System Log Explorer

CHAPTER 19

Attaching Multiple Firewall Filters to a Single Interface

IN THIS CHAPTER

- Applying Firewall Filters to Interfaces | 1300
- Configuring Firewall Filters | 1301
- Multifield Classifier Example: Configuring Multifield Classification | 1304
- Multifield Classifier for Ingress Queuing on MX Series Routers with MPC | 1330
- Assigning Multifield Classifiers in Firewall Filters to Specify Packet-Forwarding Behavior (CLI Procedure) | 1331
- Understanding Multiple Firewall Filters in a Nested Configuration | 1334
- Guidelines for Nesting References to Multiple Firewall Filters | 1336
- Understanding Multiple Firewall Filters Applied as a List | 1338
- Guidelines for Applying Multiple Firewall Filters as a List | 1342
- Example: Applying Lists of Multiple Firewall Filters | 1344
- Example: Nesting References to Multiple Firewall Filters | 1352
- Example: Filtering Packets Received on an Interface Set | 1358

Applying Firewall Filters to Interfaces

For a firewall filter to work, you must apply it to at least one Layer 3 interface. To do this, include the `filter` statement when configuring a logical interface at the `[edit interfaces]` hierarchy level:

```
[edit interfaces]
user@switch# set interface-name unit logical-unit-number family (inet | inet6) filter (input |
output) filter-name
```

In the `input` statement, specify a firewall filter to be evaluated when packets are received on the interface. Input filters applied to a loopback interface affect only traffic destined for the Routing Engine.

In the output statement, specify a filter to be evaluated when packets exit the interface.

NOTE: When you create a loopback interface, it is important to apply an ingress filter to it so the Routing Engine is protected. We recommend that when you apply a filter to the loopback interface lo0, you include the apply-groups statement. Doing so ensures that the filter is automatically inherited on every loopback interface, including lo0 and other loopback interfaces.

RELATED DOCUMENTATION

Configuring Firewall Filters | 1826

Configuring Firewall Filters

IN THIS SECTION

- Configuring a Firewall Filter | 1301
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | 1303

You can configure firewall filters in a switch to control traffic that enters or exits Layer 3 (routed) interfaces. To use a firewall filter, you must configure the filter and then apply it to a Layer 3 interface.

Configuring a Firewall Filter

To configure a firewall filter:

1. Configure the family address type, filter name, term name, and at least one match condition—for example, match on packets that contain a specific source address:

```
[edit]
user@switch# set firewall family (inet | inet6) filter ingress-port-filter term t1 from
source-address 192.0.2.14
```

   Specify the family address type inet for IPv4 or inet6 for IPv6.

The filter and term names can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. Each filter name must be unique. A filter can contain one or more terms, and each term name must be unique within a filter.

2. Configure additional match conditions. For example, match on packets that contain a specific source port:

```
[edit firewall family inet filter ingress-port-filter term t1 from]
user@switch# set source-port 80
```

You can specify one or more match conditions in a single `from` statement. For a match to occur, the packet must match all the conditions in the term. The `from` statement is optional, but if included in a term, it cannot be empty. If you omit the `from` statement, all packets are considered to match.

3. If you want to apply a firewall filter to multiple interfaces and be able to see counters specific to each interface, configure the `interface-specific` option:

```
[edit firewall family inet filter ingress-port-filter]
user@switch# set interface-specific
```

4. In each firewall filter term, specify the actions to take if the packet matches all the conditions in that term. You can specify an action and action modifiers:

   - To specify a filter action, for example, to discard packets that match the conditions of the filter term:

     ```
     [edit firewall family inet filter ingress-port-filter term t1 then]
     user@switch# set discard
     ```

     You can specify no more than one action (`accept`, `discard`, `reject`, `routing-instance`, or `vlan`) per term.

   - To specify action modifiers, for example, to count and classify packets to a forwarding class. For example:

     ```
     [edit firewall family inet filter ingress-port-filter term t1 then]
     user@switch# set count counter-one
     user@switch# set loss-priority high
     ```

If you omit the `then` statement or do not specify an action, packets that match all the conditions in the `from` statement are accepted. However, you should always explicitly configure an action in the `then` statement. You can include no more than one action statement, but you can use any combination of

action modifiers. For an action or action modifier to take effect, all conditions in the `from` statement must match.

NOTE: Implicit discard is also applicable to a firewall filter applied to the loopback interface, `lo0`.

NOTE: For the complete list of match conditions, actions, and action modifiers, see Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700). Note that on the OCX1100 switch you can use only those match conditions that are valid for IPv4 and IPv6 interfaces.

Applying a Firewall Filter to a Layer 3 (Routed) Interface

To apply a firewall filter to a Layer 3 interface:

1. Provide a meaningful description of the firewall filter in the configuration of the interface to which the filter will be applied:

```
[edit]
user@switch# set interfaces xe-0/0/1 description "filter to count and monitor traffic on
layer 3 interface"
```

2. You can apply firewall filters to filter packets that enter or exit a Layer 3 interface:
   - To apply a firewall filter to filter packets that enter a Layer 3 interface:

```
[edit]
user@switch# set interfaces xe-0/0/1 unit 0 family inet filter input ingress-router-filter
```

   - To apply a firewall filter to filter packets that exit a Layer 3 interface:

```
[edit]
user@switch# set interfaces xe-0/0/2 unit 0 family inet filter output egress-router-filter
```

NOTE: You can apply only one filter to an interface for a given direction (ingress or egress).

RELATED DOCUMENTATION

Overview of Firewall Filters (OCX Series) | 846

Monitoring Firewall Filter Traffic | 827

Configuring Port Mirroring

Multifield Classifier Example: Configuring Multifield Classification

IN THIS SECTION

- Multifield Classification Overview | 1304
- Multifield Classification Requirements and Restrictions | 1307
- Multifield Classification Limitations on M Series Routers | 1308
- Example: Configuring Multifield Classification | 1311
- Example: Configuring and Applying a Firewall Filter for a Multifield Classifier | 1321

Multifield Classification Overview

IN THIS SECTION

- Forwarding Classes and PLP Levels | 1304
- Multifield Classification and BA Classification | 1305
- Multifield Classification Used In Conjunction with Policers | 1306

Forwarding Classes and PLP Levels

You can configure the Junos OS class of service (CoS) features to classify incoming traffic by associating each packet with a forwarding class, a packet loss priority (PLP) level, or both:

- Based on the associated forwarding class, each packet is assigned to an output queue, and the router services the output queues according to the associated scheduling you configure.

- Based on the associated PLP, each packet carries a lower or higher likelihood of being dropped if congestion occurs. The CoS random early detection (RED) process uses the drop probability configuration, output queue fullness percentage, and packet PLP to drop packet as needed to control congestion at the output stage.

Multifield Classification and BA Classification

The Junos OS supports two general types of packet classification: behavior aggregate (BA) classification and multifield classification:

- BA classification, or CoS value traffic classification, refers to a method of packet classification that uses a CoS configuration to set the forwarding class or PLP of a packet based on the *CoS value* in the IP packet header. The CoS value examined for BA classification purposes can be the Differentiated Services code point (DSCP) value, DSCP IPv6 value, IP precedence value, MPLS EXP bits, and IEEE 802.1p value. The default classifier is based on the IP precedence value.

- Multifield classification refers to a method of packet classification that uses a standard stateless *firewall filter* configuration to set the forwarding class or PLP for each packet entering or exiting the interface based on *multiple fields* in the IP packet header, including the DSCP value (for IPv4 only), the IP precedence value, the MPLS EXP bits, and the IEEE 802.1p bits. Multifield classification commonly matches on IP address fields, the IP protocol type field, or the port number in the UDP or TCP pseudoheader field. Multifield classification is used instead of BA classification when you need to classify packets based on information in the packet information other than the CoS values only.

  With multifield classification, a firewall filter term can specify the packet classification actions for matching packets though the use of the `forwarding-class` *`class-name`* or `loss-priority (high | medium-high | medium-low | low)` nonterminating actions in the term's `then` clause.

NOTE: BA classification of a packet can be overridden by the stateless firewall filter actions `forwarding-class` and `loss-priority`.

NOTE: Misclassification of traffic via Multifield classifier on QFX5k:

- If BUM traffic is forced to take unicast queue via MF classifier, packets will be classified to MCQ9. Junos releases 21.4R3 , 22.1R3 and from Junos release version 22.2 these packets will be classified to MCQ8.

- If unicast traffic is forced to take multicast queue via MF classifier, packets will be classified to MCQ9 and from release 19.1R3 it will be classified to best-effort queue.

Multifield Classification Used In Conjunction with Policers

To configure multifield classification in conjunction with rate limiting, a firewall filter term can specify the packet classification actions for matching packets through the use of a `policer` nonterminating action that references a single-rate two-color policer.

When multifield classification is configured to perform classification through a policer, the filter-matched packets in the traffic flow are rate-limited to the policer-specified traffic limits. Packets in a conforming flow of filter-matched packets are implicitly set to a `low` PLP. Packets in a nonconforming traffic flow can be discarded, or the packets can be set to a specified forwarding class, set to a specified PLP level, or both, depending on the type of policer and how the policer is configured to handle nonconforming traffic.

NOTE: Before you apply a firewall filter that performs multifield classification and also a policer to the same *logical interface* and for the same traffic direction, make sure that you consider the order of policer and firewall filter operations.

As an example, consider the following scenario:

- You configure a firewall filter that performs multifield classification (acts on matched packets by setting the forwarding class, the PLP, or both) based on the packet's existing forwarding class or PLP. You apply the firewall filter at the input of a logical interface.
- You also configure a single-rate two-color policer that acts on a red traffic flow by re-marking (setting the forwarding class, the PLP, or both) rather than discarding those packets. You apply the policer as an interface policer at the input of the same logical interface to which you apply the firewall filter.

Because of the order of policer and firewall operations, the input policer is executed before the input firewall filter. This means that the multifield classification specified by the firewall filter is performed on input packets that have already been re-marked once by policing actions. Consequently, any input packet that matches the conditions specified in a firewall filter term is then subject to a second re-marking according to the `forwarding-class` or `loss-priority` nonterminating actions also specified in that term.

SEE ALSO

Firewall Filter Nonterminating Actions | **871**

Junos OS Routing Policies, Firewall Filters and Traffic Policers User Guide for Routing Devices

Order of Policer and Firewall Filter Operations | **1926**

Two-Color Policer Configuration Overview | **2034**

Multifield Classification Requirements and Restrictions
Multifield Classification Limitations on M Series Routers
Example: Configuring Multifield Classification
The Junos OS CoS Components Used to Manage Congestion and Control Service Levels
Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic
Understanding How Forwarding Classes Assign Classes to Output Queues
Default Forwarding Classes
Managing Congestion Using RED Drop Profiles and Packet Loss Priorities

Multifield Classification Requirements and Restrictions

IN THIS SECTION

- Supported Platforms | **1307**
- CoS Tricolor Marking Requirement | **1307**
- Restrictions | **1308**

Supported Platforms

The `loss-priority` *firewall filter* action is supported on the following routing platforms only:

- EX Series switches
- M7i and M10i routers with the Enhanced CFEB (CFEB-E)
- M120 and M320 routers
- MX Series routers
- T Series routers with Enhanced II Flexible PIC Concentrators (FPCs)
- PTX Series routers

CoS Tricolor Marking Requirement

The `loss-priority` firewall filter action has platform-specific requirements dependencies on the CoS tricolor marking feature, as defined in RFC 2698:

- On an M320 router, you cannot commit a configuration that includes the `loss-priority` firewall filter action unless you enable the CoS tricolor marking feature.

- On all routing platforms that support the `loss-priority` firewall filter action, you cannot set the `loss-priority` firewall filter action to `medium-low` or `medium-high` unless you enable the CoS tricolor marking feature. .

To enable the CoS tricolor marking feature, include the `tri-color` statement at the `[edit class-of-service]` hierarchy level.

Restrictions

You cannot configure the `loss-priority` and `three-color-policer` nonterminating actions for the same firewall filter term. These two nonterminating actions are mutually exclusive.

NOTE: On a PTX Series router, you must configure the `policer` action in a separate rule and not combine it with the rule configuring the `forwarding-class`, and `loss-priority` actions. See "Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers" on page 1823.

SEE ALSO

Firewall Filter Nonterminating Actions | 871

Two-Color Policer Configuration Overview | 2034

Multifield Classification Overview

Multifield Classification Limitations on M Series Routers

Example: Configuring Multifield Classification

tri-color

Multifield Classification Limitations on M Series Routers

IN THIS SECTION

- Problem: Output-Filter Matching on Input-Filter Classification | 1309
- Workaround: Configure All Actions in the Ingress Filter | 1310

Problem: Output-Filter Matching on Input-Filter Classification

On M Series routers (except M120 routers), you cannot classify packets with an output filter match based on the ingress classification that is set with an input filter applied to the same IPv4 *logical interface*.

For example, in the following configuration, the filter called `ingress` assigns all incoming IPv4 packets to the `expedited-forwarding` class. The filter called `egress` counts all packets that were assigned to the `expedited-forwarding` class in the `ingress` filter. This configuration does not work on most M Series routers. It works on all other routing platforms, including M120 routers, MX Series routers, and T Series routers.

```
[edit]
user@host # show firewall
family inet {
    filter ingress {
        term 1 {
            then {
                forwarding-class expedited-forwarding;
                accept;
            }
        }
        term 2 {
            then accept;
        }
    }
    filter egress {
        term 1 {
            from {
                forwarding-class expedited-forwarding;
            }
            then count ef;
        }
        term 2 {
            then accept;
        }
    }
}

[edit]
user@host# show interfaces
ge-1/2/0 {
    unit 0 {
```

```
        family inet {
            filter {
                input ingress;
                output egress;
            }
        }
    }
}
```

Workaround: Configure All Actions in the Ingress Filter

As a workaround, you can configure all of the actions in the ingress filter.

```
user@host # show firewall
family inet {
    filter ingress {
        term 1 {
            then {
                forwarding-class expedited-forwarding;
                accept;
                count ef;
            }
        }
        term 2 {
            then accept;
        }
    }
}

[edit]
user@host# show interfaces
ge-1/2/0 {
    unit 0 {
        family inet {
            filter {
                input ingress;
            }
        }
    }
}
```

SEE ALSO

Two-Color Policer Configuration Overview | 2034

Multifield Classification Overview

Multifield Classification Requirements and Restrictions

Example: Configuring Multifield Classification

Example: Configuring Multifield Classification

IN THIS SECTION

- Requirements | 1311
- Overview | 1312
- Configuration | 1313
- Verification | 1319

This example shows how to configure multifield classification of IPv4 traffic by using firewall filter actions and two firewall filter policers.

Requirements

Before you begin, make sure that your environment supports the features shown in this example:

1. The `loss-priority` firewall filter action must be supported on the router and configurable to all four values.

   a. To be able to set a `loss-priority` firewall filter action, configure this example on logical interface `ge-1/2/0.0` on one of the following routing platforms:

      - MX Series router
      - M120 or M320 router
      - M7i or M10i router with the Enhanced CFEB (CFEB-E)
      - T Series router with Enhanced II Flexible PIC Concentrator (FPC)

   b. To be able to set a `loss-priority` firewall filter action to `medium-low` or `medium-high`, make sure that the CoS tricolor marking feature is enabled. To enable the CoS tricolor marking feature, include the `tri-color` statement at the `[edit class-of-service]` hierarchy level.

2. The `expedited-forwarding` and `assured-forwarding` forwarding classes must be scheduled on the underlying physical interface `ge-1/2/0`.

   a. Make sure that the following forwarding classes are assigned to output queues:

      - `expedited-forwarding`
      - `assured-forwarding`

      Forwarding-class assignments are configured at the `[edit class-of-service forwarding-classes queue` *`queue-number`*`]` hierarchy level.

      > **NOTE**: You cannot commit a configuration that assigns the same forwarding class to two different queues.

   b. Make sure that the output queues to which the forwarding classes are assigned are associated with schedulers. A scheduler defines the amount of interface bandwidth assigned to the queue, the size of the memory buffer allocated for storing packets, the priority of the queue, and the random early detection (RED) drop profiles associated with the queue.

      - You configure output queue schedulers at the `[edit class-of-service schedulers]` hierarchy level.
      - You associate output queue schedulers with forwarding classes by means of a scheduler map that you configure at the `[edit class-of-service scheduler-maps` *`map-name`*`]` hierarchy level.

   c. Make sure that output-queue scheduling is applied to the physical interface `ge-1/2/0`.

      You apply a scheduler map to a physical interface at the `[edit class-of-service interfaces ge-1/2/0 scheduler-map` *`map-name`*`]` hierarchy level.

Overview

IN THIS SECTION

- Topology | **1313**

In this example, you apply multifield classification to the input IPv4 traffic at a logical interface by using stateless firewall filter actions and two firewall filter policers that are referenced from the firewall filter. Based on the source address field, packets are either set to the `low` loss priority or else policed. Neither of the policers discards nonconforming traffic. Packets in nonconforming flows are marked for a specific

forwarding class (expedited-forwarding or assured-forwarding), set to a specific loss priority, and then transmitted.

NOTE: Single-rate two-color policers always transmit packets in a conforming traffic flow after implicitly setting a low loss priority.

Topology

In this example, you apply multifield classification to the IPv4 traffic on logical interface ge-1/2/0.0. The classification rules are specified in the IPv4 stateless firewall filter mfc-filter and two single-rate two-color policers, ef-policer and af-policer.

The IPv4 standard stateless firewall filter mfc-filter defines three filter terms:

- isp1-customers—The first filter term matches packets with the source address 10.1.1.0/24 or 10.1.2.0/24. Matched packets are assigned to the expedited-forwarding forwarding class and set to the low loss priority.
- isp2-customers—The second filter term matches packets with the source address 10.1.3.0/24 or 10.1.4.0/24. Matched packets are passed to ef-policer, a policer that rate-limits traffic to a bandwidth limit of 300 Kbps with a burst-size limit of 50 KB. This policer specifies that packets in a nonconforming flow are marked for the expedited-forwarding forwarding class and set to the high loss priority.
- other-customers—The third and final filter term passes all other packets to af-policer, a policer that rate-limits traffic to a bandwidth limit of 300 Kbps and a burst-size limit of 50 KB (the same traffic limits as defined by ef-policer). This policer specifies that packets in a nonconforming flow are marked for the assured-forwarding forwarding class and set to the medium-high loss priority.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1314**
- Configuring Policers to Rate-Limit Expedited-Forwarding and Assured-Forwarding Traffic | **1314**
- Configuring a Multifield Classification Filter That Also Applies Policing | **1316**
- Applying Multifield Classification Filtering and Policing to the Logical Interface | **1318**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall policer ef-policer if-exceeding bandwidth-limit 300k
set firewall policer ef-policer if-exceeding burst-size-limit 50k
set firewall policer ef-policer then loss-priority high
set firewall policer ef-policer then forwarding-class expedited-forwarding
set firewall policer af-policer if-exceeding bandwidth-limit 300k
set firewall policer af-policer if-exceeding burst-size-limit 50k
set firewall policer af-policer then loss-priority high
set firewall policer af-policer then forwarding-class assured-forwarding
set firewall family inet filter mfc-filter term isp1-customers from source-address 10.1.1.0/24
set firewall family inet filter mfc-filter term isp1-customers from source-address 10.1.2.0/24
set firewall family inet filter mfc-filter term isp1-customers then loss-priority low
set firewall family inet filter mfc-filter term isp1-customers then forwarding-class expedited-
forwarding
set firewall family inet filter mfc-filter term isp2-customers from source-address 10.1.3.0/24
set firewall family inet filter mfc-filter term isp2-customers from source-address 10.1.4.0/24
set firewall family inet filter mfc-filter term isp2-customers then policer ef-policer
set firewall family inet filter mfc-filter term other-customers then policer af-policer
set interfaces ge-1/2/0 unit 0 family inet address 192.168.1.1/24
set interfaces ge-1/2/0 unit 0 family inet filter input mfc-filter
```

Configuring Policers to Rate-Limit Expedited-Forwarding and Assured-Forwarding Traffic

Step-by-Step Procedure

To configure policers to rate-limit expedited-forwarding and assured-forwarding traffic:

1. Define traffic limits for expedited-forwarding traffic.

```
[edit]
user@host# edit firewall policer ef-policer
```

```
[edit firewall policer ef-policer]
user@host# set if-exceeding bandwidth-limit 300k
user@host# set if-exceeding burst-size-limit 50k
user@host# set then loss-priority high
user@host# set then forwarding-class expedited-forwarding
```

2. Configure a policer for assured-forwarding traffic.

```
[edit firewall policer ef-policer]
user@host# up

[edit firewall]
user@host# edit policer af-policer

[edit firewall policer af-policer]
user@host# set if-exceeding bandwidth-limit 300k
user@host# set if-exceeding burst-size-limit 50k
user@host# set then loss-priority high
user@host# set then forwarding-class assured-forwarding
```

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer af-policer {
    if-exceeding {
        bandwidth-limit 300k;
        burst-size-limit 50k;
    }
    then {
        loss-priority high;
        forwarding-class assured-forwarding;
    }
}
policer ef-policer {
    if-exceeding {
```

```
        bandwidth-limit 300k;
        burst-size-limit 50k;
    }
    then {
        loss-priority high;
        forwarding-class expedited-forwarding;
    }
}
```

Configuring a Multifield Classification Filter That Also Applies Policing

Step-by-Step Procedure

To configure a multifield classification filter that additionally applies policing:

1. Enable configuration of a firewall filter term for IPv4 traffic.

```
[edit]
user@host# edit firewall family inet filter mfc-filter
```

2. Configure the first term to match on source addresses and then classify the matched packets.

```
[edit firewall family inet filter mfc-filter]
user@host# set term isp1-customers from source-address 10.1.1.0/24
user@host# set term isp1-customers from source-address 10.1.2.0/24
user@host# set term isp1-customers then loss-priority low
user@host# set term isp1-customers then forwarding-class expedited-forwarding
```

3. Configure the second term to match on different source addresses and then police the matched packets.

```
[edit firewall family inet filter mfc-filter]
user@host# set term isp2-customers from source-address 10.1.3.0/24
user@host# set term isp2-customers from source-address 10.1.4.0/24
user@host# set term isp2-customers then policer ef-policer
```

4. Configure the third term to police all other packets to a different set of traffic limits and actions.

```
[edit firewall family inet filter mfc-filter]
user@host# set term other-customers then policer af-policer
```

Results

Confirm the configuration of the filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter mfc-filter {
        term isp1-customers {
            from {
                source-address 10.1.1.0/24;
                source-address 10.1.2.0/24;
            }
            then {
                loss-priority low;
                forwarding-class expedited-forwarding;
            }
        }
        term isp2-customers {
            from {
                source-address 10.1.3.0/24;
                source-address 10.1.4.0/24;
            }
            then {
                policer ef-policer;
            }
        }
        term other-customers {
            then {
                policer af-policer;
            }
        }
    }
}
```

```
policer af-policer {
    if-exceeding {
        bandwidth-limit 300k;
        burst-size-limit 50k;
    }
    then discard;
}
policer ef-policer {
    if-exceeding {
        bandwidth-limit 200k;
        burst-size-limit 50k;
    }
    then {
        loss-priority high;
        forwarding-class expedited-forwarding;
    }
}
```

Applying Multifield Classification Filtering and Policing to the Logical Interface

Step-by-Step Procedure

To apply multifield classification filtering and policing to the logical interface:

1. Enable configuration of IPv4 on the logical interface.

```
[edit]
user@host# edit interfaces ge-1/2/0 unit 0 family inet
```

2. Configure an IP address for the logical interface.

```
[edit interfaces ge-1/2/0 unit 0 family inet ]
user@host# set address 192.168.1.1/24
```

3. Apply the firewall filter to the logical interface input.

```
[edit interfaces ge-1/2/0 unit 0 family inet ]
user@host# set filter input mfc-filter
```

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.

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/2/0 {
    unit 0 {
        family inet {
            filter {
                input mfc-filter;
            }
            address 192.168.1.1/24;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Number of Packets Processed by the Policer at the Logical Interface | 1320

Confirm that the configuration is working properly.

Displaying the Number of Packets Processed by the Policer at the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show firewall` operational mode command for the filter you applied to the logical interface.

```
user@host> show firewall filter rate-limit-in
Filter: rate-limit-in
Policers:
Name                                                Packets
ef-policer-isp2-customers                             32863
af-policer-other-customers                             3870
```

The command output lists the policers applied by the firewall filter `rate-limit-in`, and the number of packets that matched the filter term.

NOTE: The packet count includes the number of out-of-specification (out-of-spec) packet counts, not all packets policed by the policer.

The policer name is displayed concatenated with the name of the firewall filter term in which the policer is referenced as an action.

SEE ALSO

Two-Color Policer Configuration Overview | 2034

Multifield Classification Overview

Multifield Classification Requirements and Restrictions

Multifield Classification Limitations on M Series Routers

tri-color

Example: Configuring and Applying a Firewall Filter for a Multifield Classifier

IN THIS SECTION

- Requirements | 1321
- Overview | 1321
- Configuration | 1323
- Verification | 1327

This example shows how to configure a firewall filter to classify traffic using a multifield classifier. The classifier detects packets of interest to class of service (CoS) as they arrive on an interface. Multifield classifiers are used when a simple behavior aggregate (BA) classifier is insufficient to classify a packet, when peering routers do not have CoS bits marked, or the peering router's marking is untrusted.

Requirements

To verify this procedure, this example uses a traffic generator. The traffic generator can be hardware-based or it can be software running on a server or host machine.

The functionality in this procedure is widely supported on devices that run Junos OS. The example shown here was tested and verified on MX Series routers running Junos OS Release 10.4.

Overview

IN THIS SECTION

- Topology | 1322

A classifier is a software operation that inspects a packet as it enters the router or switch. The packet header contents are examined, and this examination determines how the packet is treated when the network becomes too busy to handle all of the packets and you want your devices to drop packets intelligently, instead of dropping packets indiscriminately. One common way to detect packets of interest is by source port number. The TCP port numbers 80 and 12345 are used in this example, but many other matching criteria for packet detection are available to multifield classifiers, using firewall filter match conditions. The configuration in this example specifies that TCP packets with source port 80

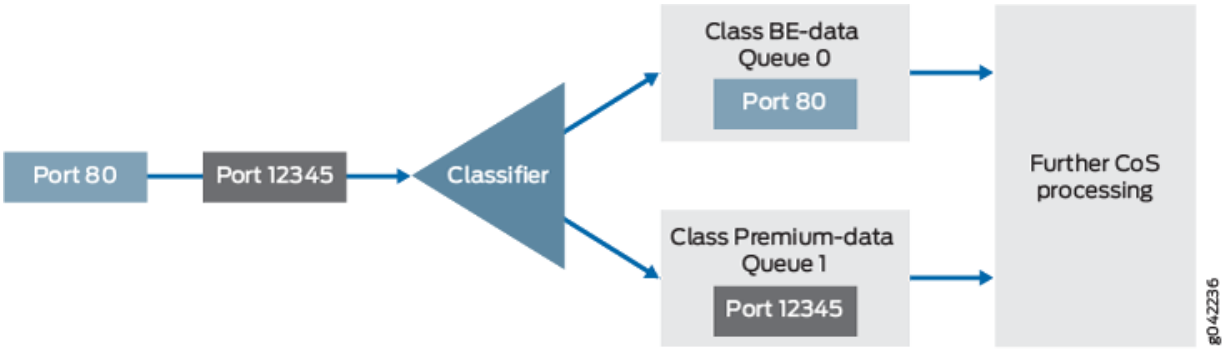
are classified into the BE-data forwarding class and queue number 0. TCP packets with source port 12345 are classified into the Premium-data forwarding class and queue number 1.

Multifield classifiers are typically used at the network edge as packets enter an autonomous system (AS).

In this example, you configure the firewall filter mf-classifier and specify some custom forwarding classes on Device R1. In specifying the custom forwarding classes, you also associate each class with a queue.

The classifier operation is shown in Figure 58 on page 1322.

Figure 58: Multifield Classifier Based on TCP Source Ports

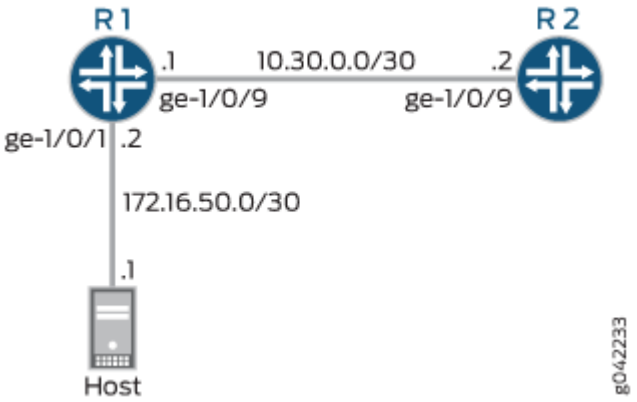


You apply the multifield classifier's firewall filter as an input filter on each customer-facing or host-facing interface that needs the filter. The incoming interface is ge-1/0/1 on Device R1. The classification and queue assignment is verified on the outgoing interface. The outgoing interface is Device R1's ge-1/0/9 interface.

Topology

Figure 59 on page 1323 shows the sample network.

Figure 59: Multifield Classifier Scenario



"CLI Quick Configuration" on page 1323 shows the configuration for all of the Juniper Networks devices in Figure 59 on page 1323.

"Step-by-Step Procedure" on page 1324 describes the steps on Device R1.

Classifiers are described in more detail in the following Juniper Networks Learning Byte video.

Video: Class of Service Basics, Part 2: Classification Learning Byte

Configuration

IN THIS SECTION

- Procedure | **1323**

Procedure

CLI Quick Configuration

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

Device R1

```
set interfaces ge-1/0/1 description to-host
set interfaces ge-1/0/1 unit 0 family inet filter input mf-classifier
set interfaces ge-1/0/1 unit 0 family inet address 172.16.50.2/30
set interfaces ge-1/0/9 description to-R2
set interfaces ge-1/0/9 unit 0 family inet address 10.30.0.1/30
set class-of-service forwarding-classes class BE-data queue-num 0
set class-of-service forwarding-classes class Premium-data queue-num 1
set class-of-service forwarding-classes class Voice queue-num 2
set class-of-service forwarding-classes class NC queue-num 3
set firewall family inet filter mf-classifier term BE-data from protocol tcp
set firewall family inet filter mf-classifier term BE-data from port 80
set firewall family inet filter mf-classifier term BE-data then forwarding-class BE-data
set firewall family inet filter mf-classifier term Premium-data from protocol tcp
set firewall family inet filter mf-classifier term Premium-data from port 12345
set firewall family inet filter mf-classifier term Premium-data then forwarding-class Premium-
data
set firewall family inet filter mf-classifier term accept-all-else then accept
```

Device R2

```
set interfaces ge-1/0/9 description to-R1
set interfaces ge-1/0/9 unit 0 family inet address 10.30.0.2/30
```

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set ge-1/0/1 description to-host
user@R1# set ge-1/0/1 unit 0 family inet address 172.16.50.2/30
user@R1# set ge-1/0/9 description to-R2
user@R1# set ge-1/0/9 unit 0 family inet address 10.30.0.1/30
```

2. Configure the custom forwarding classes and associated queue numbers.

```
[edit class-of-service forwarding-classes]
user@R1# set BE-data queue-num 0
user@R1# set Premium-data queue-num 1
user@R1# set Voice queue-num 2
user@R1# set NC queue-num 3
```

3. Configure the firewall filter term that places TCP traffic with a source port of 80 (HTTP traffic) into the BE-data forwarding class, associated with queue 0.

```
[edit firewall family inet filter mf-classifier]
user@R1# set term BE-data from protocol tcp
user@R1# set term BE-data from port 80
user@R1# set term BE-data then forwarding-class BE-data
```

4. Configure the firewall filter term that places TCP traffic with a source port of 12345 into the Premium-data forwarding class, associated with queue 1.

```
[edit firewall family inet filter mf-classifier]
user@R1# set term Premium-data from protocol tcp
user@R1# set term Premium-data from port 12345
user@R1# set term Premium-data then forwarding-class Premium-data
```

5. At the end of your firewall filter, configure a default term that accepts all other traffic.

   Otherwise, all traffic that arrives on the interface and is not explicitly accepted by the firewall filter is discarded.

```
[edit firewall family inet filter mf-classifier]
user@R1# set term accept-all-else then accept
```

6. Apply the firewall filter to the ge-1/0/1 interface as an input filter.

```
[edit interfaces]
user@R1# set ge-1/0/1 unit 0 family inet filter input mf-classifier
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show class-of-service`, `show firewall` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
ge-1/0/1 {
    description to-host;
    unit 0 {
        family inet {
            filter {
                input mf-classifier;
            }
            address 172.16.50.2/30;
        }
    }
}
ge-1/0/9 {
    description to-R2;
    unit 0 {
        family inet {
            address 10.30.0.1/30;
        }
    }
}
```

```
user@R1# show class-of-service
forwarding-classes {
    class BE-data queue-num 0;
    class Premium-data queue-num 1;
    class Voice queue-num 2;
    class NC queue-num 3;
}
```

```
user@R1# show firewall
family inet {
    filter mf-classifier {
        term BE-data {
```

```
            from {
                protocol tcp;
                port 80;
            }
            then forwarding-class BE-data;
        }
        term Premium-data {
            from {
                protocol tcp;
                port 12345;
            }
            then forwarding-class Premium-data;
        }
        term accept-all-else {
            then accept;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Checking the CoS Settings | **1327**
- Sending TCP Traffic into the Network and Monitoring the Queue Placement | **1328**

Confirm that the configuration is working properly.

Checking the CoS Settings

Purpose

Confirm that the forwarding classes are configured correctly.

Action

From Device R1, run the `show class-of-service forwardng-classes` command.

```
user@R1> show class-of-service forwarding-class

Forwarding class                      ID      Queue  Restricted queue  Fabric priority
Policing priority   SPU priority
  BE-data                              0       0          0             low
normal            low
  Premium-data                         1       1          1             low
normal            low
  Voice                                2       2          2             low
normal            low
  NC                                   3       3          3             low
normal            low
```

Meaning

The output shows the configured custom classifier settings.

Sending TCP Traffic into the Network and Monitoring the Queue Placement

Purpose

Make sure that the traffic of interest is sent out the expected queue.

Action

1. Clear the interface statistics on Device R1's outgoing interface.

   ```
   user@R1> clear interfaces statistics ge-1/0/9
   ```

2. Use a traffic generator to send 50 TCP port 80 packets to Device R2 or to some other downstream device.

3. On Device R1, check the queue counters.

Notice that you check the queue counters on the downstream output interface, not on the incoming interface.

```
user@R1> show interfaces extensive ge-1/0/9 | find "Queue counters"

  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0                              50                   50                    0
    1                               0                   57                    0
    2                               0                    0                    0
    3                               0                    0                    0
```

4. Use a traffic generator to send 50 TCP port 12345 packets to Device R2 or to some other downstream device.

```
[root@host]#  hping 172.16.60.1  -c 50 -s 12345  -k
```

5. On Device R1, check the queue counters.

```
user@R1> show interfaces extensive ge-1/0/9 | find "Queue counters"

  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0                              50                   50                    0
    1                              50                   57                    0
    2                               0                    0                    0
    3                               0                    0                    0
```

Meaning

The output shows that the packets are classified correctly. When port 80 is used in the TCP packets, queue 0 is incremented. When port 12345 is used, queue 1 is incremented.

SEE ALSO

Example: Configuring a Two-Rate Three-Color Policer | 2157

RELATED DOCUMENTATION

- Firewall Filter Nonterminating Actions | 871
- Order of Policer and Firewall Filter Operations | 1926
- Two-Color Policer Configuration Overview | 2034
- Guidelines for Applying Traffic Policers | 1934
- *The Junos OS CoS Components Used to Manage Congestion and Control Service Levels*
- *Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic*
- *Understanding How Forwarding Classes Assign Classes to Output Queues*
- *Default Forwarding Classes*
- *Managing Congestion Using RED Drop Profiles and Packet Loss Priorities*
- *tri-color*

Multifield Classifier for Ingress Queuing on MX Series Routers with MPC

Beginning with Junos OS Release 16.1, the multifield classifier for ingress queuing is an implementation point for firewall filters configured with specific traffic shaping actions. These filters allow you to set the forwarding class and packet loss priority for packets, or drop the packets prior to ingress queue selection. The filters are applied as ingress queue filters. Class-of-service (CoS) commands can then be used to select ingress queue, set rate limiting and so forth.

Firewall filters configured at the protocol family level are able to distinguish specific types of traffic from other types by matching on multiple fields within the packet header. The number and types of matches available depend on which protocol family is used in the filter. Before the introduction of the ingress queuing filter, these firewall filters could only be applied to traffic after the ingress queue had been selected based solely on the behavior aggregate (BA). With the introduction of the ingress queuing filter, firewall filters can be used to set forwarding classification and packet loss priority based on multiple fields within the packet header prior to forwarding queue selection. CoS functions provide traffic classification options and the ability to assign that classified traffic to specific forwarding queues.

NOTE: Ingress queuing filters are only available when the traffic manager mode is set to `ingress-and-egress` at the `[edit chassis fpc` *`fpc-id`* `pic` *`pic-id`* `traffic-manager mode]` hierarchy level.

The `ingress-queuing-filter` configuration statement is used at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family` *`family-name`*`]` hierarchy level to designate a previously configured firewall filter to be used as an ingress queuing filter. The following list shows which protocol families are compatible with the `ingress-queuing-filter` statement:

- bridge
- ccc
- inet
- inet6
- mpls
- vpls

The named firewall filter is a normal firewall filter that must be configured with at least one of the following actions: accept, discard, forwarding-class, and loss-priority.

Release History Table

Release	Description
16.1	Beginning with Junos OS Release 16.1, the multifield classifier for ingress queuing is an implementation point for firewall filters configured with specific traffic shaping actions.

RELATED DOCUMENTATION

Understanding How Behavior Aggregate Classifiers Prioritize Trusted Traffic

ingress-queuing-filter

Example: Configuring a Filter for Use as an Ingress Queuing Filter | 1138

Assigning Multifield Classifiers in Firewall Filters to Specify Packet-Forwarding Behavior (CLI Procedure)

You can configure firewall filters with multifield classifiers to classify packets transiting a port, VLAN, or Layer 3 interface on an EX Series switch.

You specify multifield classifiers in a firewall filter configuration to set the forwarding class and packet loss priority (PLP) for incoming or outgoing packets. By default, the data traffic that is not classified is assigned to the **best-effort** class associated with queue 0.

You can specify any of the following default forwarding classes:

Forwarding class	Queue
best-effort	0
assured-forwarding	1
expedited-forwarding	5
network-control	7

To assign multifield classifiers in firewall filters:

1. Configure the family name and filter name for the filter at the `[edit firewall]` hierarchy level, for example:

```
[edit firewall]
user@switch# set family ethernet-switching
user@switch# set family ethernet-switching filter ingress-filter
```

2. Configure the terms of the filter, including the **forwarding-class** and **loss-priority** action modifiers as appropriate. When you specify a forwarding class you must also specify the packet loss priority. For example, each of the following terms examines different packet header fields and assigns an appropriate classifier and the packet loss priority:

   - The term **voice-traffic** matches packets on the **voice-vlan** and assigns the forwarding class **expedited-forwarding** and packet loss priority **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term voice-traffic from vlan-id voice-vlan
user@switch# set term voice-traffic then forwarding-class expedited-forwarding
user@switch# set term voice-traffic then loss-priority low
```

   - The term **data-traffic** matches packets on **employee-vlan** and assigns the forwarding class **assured-forwarding** and packet loss priority **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term data-traffic from vlan-id employee-vlan
```

```
user@switch# set term data-traffic then forwarding-class assured-forwarding
user@switch# set term data-traffic then loss-priority low
```

- Because loss of network-generated packets can jeopardize proper network operation, delay is preferable to discard of packets. The following term, **network-traffic**, assigns the forwarding class **network-control** and packet loss priority **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term network-traffic from precedence net-control
user@switch# set term network-traffic then forwarding-class network
user@switch# set term network-traffic then loss-priority low
```

- The last term **accept-traffic** matches any packets that did not match on any of the preceding terms and assigns the forwarding class **best-effort** and packet loss priority **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term accept-traffic from precedence net-control
user@switch# set term accept-traffic then forwarding-class best-effort
user@switch# set term accept-traffic then loss-priority low
```

3. Apply the filter **ingress-filter** to a port, VLAN or Layer 3 interface. For information about applying the filter, see "Configuring Firewall Filters (CLI Procedure)" on page 1638.

RELATED DOCUMENTATION

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

Verifying That Firewall Filters Are Operational | **1866**

Monitoring Firewall Filter Traffic | **1867**

Defining CoS Classifiers (CLI Procedure)

Defining CoS Classifiers (J-Web Procedure)

Configuring Firewall Filters (CLI Procedure) | **1638**

Understanding Multiple Firewall Filters in a Nested Configuration

IN THIS SECTION

- The Challenge: Simplify Large-Scale Firewall Filter Administration | 1334
- A Solution: Configure Nested References to Firewall Filters | 1334
- Configuration of Nested Firewall Filters | 1335
- Application of Nested Firewall Filters to a Router or Switch Interface | 1335

The Challenge: Simplify Large-Scale Firewall Filter Administration

Typically, you apply a single *firewall filter* to an interface in the input or output direction or both. This approach might not be practical, however, when you have a router (or switch) configured with many, even hundreds of interfaces. In an environment of this scale, you want the flexibility of being able to modify filtering terms common to multiple interfaces without having to reconfigure the filter of every affected interface.

In general, the solution is to apply an effectively "chained" structure of multiple stateless firewall filters to a single interface. You partition your filtering terms into multiple firewall filters configured so that you can apply a unique filter to each router (or switch) interface but also apply common filters to multiple router (or switch) interfaces as required. The Junos OS policy framework provides two options for managing the application of multiple separate firewall filters to individual router (or switch) interfaces. One option is to apply multiple filters as a single input list or output list. The other option is to reference a stateless firewall filter from within the term of another stateless firewall filter.

A Solution: Configure Nested References to Firewall Filters

The most structured way to avoid configuring duplicate filtering terms common to multiple firewall filters is to configure multiple firewall filters so that each filter includes the shared filtering terms by *referencing* a separate filter that contains the common filtering terms. The Junos OS uses the filter terms—in the order in which they appear in the filter definition—to evaluate packets that transit the interface. If you need to modify filtering terms shared across multiple interfaces, you only need to modify one firewall filter.

NOTE: Similar to the alternative approach (applying a list of firewall filters), configuring a nested firewall filter combines multiple firewall filters into a new firewall filter definition.

Configuration of Nested Firewall Filters

Configuring a nested firewall filter for each router (or switch) interface involves separating shared packet-filtering rules from interface-specific packet-filtering rules as follows:

- For each set of packet-filtering rules common across multiple interfaces, configure a separate firewall filter that contains the shared filtering terms.
- For each router (or switch) interface, configure a separate firewall filter that contains:
  - All the filtering terms unique to that interface.
  - An additional filtering term that includes a `filter` reference to the firewall filter that contains the common filtering terms.

Application of Nested Firewall Filters to a Router or Switch Interface

Applying nested firewall filters is no different from applying an unnested firewall filter. For each interface, you can include an `input` or `output` statement (or both) within the `filter` stanza to specify the appropriate nested firewall filter.

Applying nested firewall filters to an interface, the shared filtering terms and the interface-specific firewall filters are applied through *a single nested firewall filter* that includes other filters through the `filter` statement within a separate filtering term.

NOTE: Commit check and commit do not fail for unsupported nested filters. Unsupported nested filters are the filter combinations which are are not mentioned in vty command `show jexpr dfw filter-types`.

RELATED DOCUMENTATION

Guidelines for Nesting References to Multiple Firewall Filters | 1336

Example: Nesting References to Multiple Firewall Filters | 1352

Guidelines for Nesting References to Multiple Firewall Filters

IN THIS SECTION

- Statement Hierarchy for Configuring Nested Firewall Filters | 1336
- Filter-Defining Terms and Filter-Referencing Terms | 1336
- Types of Filters Supported in Nested Configurations | 1337
- Number of Filter References in a Single Filter | 1337
- Depth of Filter Nesting | 1337

Statement Hierarchy for Configuring Nested Firewall Filters

To reference a filter from within a filter, include the `filter` *`filter-name`* statement as a separate filter term:

```
firewall firewall-name {
    family family-name {
        filter filter-name {
            term term-name {
                filter filter-name;
            }
        }
    }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- `[edit]`
- `[edit logical-systems` *`logical-system-name`*`]`

Filter-Defining Terms and Filter-Referencing Terms

You cannot configure a *firewall filter* term that both references another firewall filter and defines a match condition or action. If a firewall filter term includes the `filter` statement, then it cannot also include the `from` or `then` statement.

For example, the firewall filter term `term term1` in the configuration is *not* valid:

```
[edit]
firewall {
    family inet {
        filter filter_1 {
            term term1 {
                filter filter_2;
                from {
                    source-address 172.16.1.1/32;
                }
                then {
                    accept;
                }
            }
        }
    }
}
```

In order for `term term1` to be a valid filter term, you must either remove the `filter filter_2` statement or remove both the `from` and `then` stanzas.

Types of Filters Supported in Nested Configurations

Nested configurations of firewall filters support firewall filters only. You cannot use service filters or simple filters in a nested firewall filter configuration.

Number of Filter References in a Single Filter

The total number of filters referenced from within a filter cannot exceed 256.

Depth of Filter Nesting

The Junos OS supports a single level of firewall filter nesting. If `filter_1` references `filter_2`, you cannot configure a filter that references a filter that references `filter_1`.

RELATED DOCUMENTATION

Understanding Multiple Firewall Filters in a Nested Configuration | 1334

Example: Nesting References to Multiple Firewall Filters | 1352

Understanding Multiple Firewall Filters Applied as a List

IN THIS SECTION

- The Challenge: Simplify Large-Scale Firewall Filter Administration | 1338
- A Solution: Apply Lists of Firewall Filters | 1338
- Configuration of Multiple Filters for Filter Lists | 1339
- Application of Filter Lists to a Router Interface | 1339
- Interface-Specific Names for Filter Lists | 1339
- How Filter Lists Evaluate Packets When the Matched Term Includes Terminating or Next Term Actions | 1340
- How Filter Lists Evaluate Packets When the List Includes Protocol-Independent and IP Firewall Filters | 1342

This topic covers the following information:

The Challenge: Simplify Large-Scale Firewall Filter Administration

Typically, you apply a single *firewall filter* to an interface in the input or output direction or both. However, this approach might not be practical when you have a device configured with many interfaces. In large environments, you want the flexibility of being able to modify filtering terms common to multiple interfaces without having to reconfigure the filter of every affected interface.

In general, the solution is to apply an effectively “chained” structure of multiple firewall filters to a single interface. You partition your filtering terms into multiple firewall filters that each perform a filtering task. You can then choose which filtering tasks you want to perform for a given interface and apply the filtering tasks to that interface. In this way, you only manage the configuration for a filtering task in a single firewall filter.

The Junos OS policy framework provides two options for managing the application of multiple separate firewall filters to individual router interfaces. One option is to apply multiple filters as a single input list or output list. The other option is to reference a firewall filter from within the term of another firewall filter. This option is not supported on the PTX10003 router.

A Solution: Apply Lists of Firewall Filters

The most straightforward way to avoid configuring duplicate filtering terms common to multiple firewall filters is to configure multiple firewall filters and then apply a customized *list* of filters to each interface.

The Junos OS uses the filters—in the order in which they appear in the list—to evaluate packets that transit the interface. If you need to modify filtering terms shared across multiple interfaces, you only need to modify one firewall filter that contains those terms.

Configuration of Multiple Filters for Filter Lists

Configuring firewall filters to be applied in unique lists for each router interface involves separating shared packet-filtering rules from interface-specific packet-filtering rules as follows:

- **Unique filters**—For each set of packet-filtering rules unique to a specific interface, configure a separate firewall filter that contains only the filtering terms for that interface.
- **Shared filters**—For each set of packet-filtering rules common across two or more interfaces, consider configuring a separate firewall filter that contains the shared filtering terms.

> **TIP**: When planning for a large number firewall filters to be applied using filter lists, administrators often organize the shared filters by filtering criteria, by the services to which customers subscribe, or by the purposes of the interfaces.

Application of Filter Lists to a Router Interface

Applying a list of firewall filters to an interface is a matter of selecting the filters that meet the packet-filtering requirements of that interface. For each interface, you can include an `input-list` or `output-list` statement (or both) within the `filter` stanza to specify the relevant filters in the order in which they are to be used:

- Include any filters that contain common filtering terms relevant to the interface.
- Include the filter that contain only the filtering terms unique to the interface.

Interface-Specific Names for Filter Lists

Because a filter list is configured under an interface, the resulting concatenated filter is *interface-specific*.

> **NOTE**: When a filter list is configured under an interface, the resulting concatenated filter is interface-specific, regardless whether the firewall filters in the filter list are configured as interface-specific or not. Furthermore, the instantiation of interface-specfic firewall filters not only creates separate instances of any firewall filter counters, but also separate instances of any

policer actions. Any policers applied through an action specified in the firewall filter configuration are applied separately to each interface in the interface group.

The system-generated name of an interface-specific filter consists of the full interface name followed by either '`-i`' for an input filter list or '`-o`' for an output filter list.

- **Input filter list name**—For example, if you use the `input-list` statement to apply a chain of filters to *logical interface* `ge-1/3/0.0`, the Junos OS uses the following name for the filter:

```
ge-1/3/0.0-inet-i
```

- **Output filter list name**—For example, if you use the `output-list` statement to apply a chain of filters to logical interface `fe-0/1/2.0`, the Junos OS uses the following name for the filter:

```
fe-0/1/2.0-inet-o
```

NOTE: For Junos OS Evolved, the filter names are different. For example, if the filters are bound to the inet family, the filters are named **ge-1/3/0/0-inet-i** and **fe-0/1/2.0-inet-o**.

You can use the interface-specific name of a filter list when you enter a Junos OS *operational mode command* that specifies a firewall filter name.

How Filter Lists Evaluate Packets When the Matched Term Includes Terminating or Next Term Actions

The device evaluates a packet against the filters in a list sequentially, beginning with the first filter in the list until either a terminating action occurs or the packet is implicitly discarded.

Table 66 on page 1341 describes how a firewall filter list evaluates a packet based on whether the matched term specifies a terminating action and the `next term` action. The `next term` action is neither a terminating action nor a nonterminating action but a *flow control* action.

Table 66: Firewall Filter List Behavior

Firewall Filter Actions Included in the Matched Term		Term Description	Packet-Filtering Behavior
Terminating	next term		
Yes	—	The matched term includes a terminating action (such as `discard`) but not the `next term` action	The device executes the terminating action. No subsequent terms in the filter and no subsequent filters in the list are used to evaluate the packet.
—	Yes	The matched term includes the `next term` action, but it does not include any terminating actions.	The device executes any nonterminating actions, then the device evaluates the packet against the next term in the filter or the next filter in the list. **NOTE**: On Junos OS Evolved, `next term` cannot appear as the last term of the action. A filter term where `next term` is specified as an action but without any match conditions configured is not supported.
—	—	The matched term includes neither the `next term` action nor any terminating actions.	The device executes any nonterminating actions, then the device implicitly accepts the packet. Because the `accept` action is a terminating action, no subsequent terms in the filter and no subsequent filters in the list are used to evaluate the packet.

For information about terminating actions, see "Firewall Filter Terminating Actions" on page 884.

NOTE: You cannot configure the `next term` action with a terminating action in the same firewall filter term.

How Filter Lists Evaluate Packets When the List Includes Protocol-Independent and IP Firewall Filters

On a single interface associated with a protocol-independent (`family any`) firewall filter and a protocol-specific (`family inet` or `family inet6`) firewall filter simultaneously, the protocol-independent firewall filter executes first.

The terminating action of the first filter determines whether the second filter also evaluates the packet:

- If the first filter terminates by executing the `accept` action, the second filter doesn't evaluate the packet.
- If the first filter terminates without any terms matching the packet (an *implicit* `discard` action), the second filter also evaluates the packet.
- If the first filter terminates by executing an *explicit* `discard` action, the second filter does not evaluate the packet.

The PTX10003 router does not support a combination of protocol-independent and other filters in filter-lists.

RELATED DOCUMENTATION

How Standard Firewall Filters Evaluate Packets | **793**

Guidelines for Applying Multiple Firewall Filters as a List | **1342**

Example: Applying Lists of Multiple Firewall Filters | **1344**

Guidelines for Applying Multiple Firewall Filters as a List

IN THIS SECTION

- Statement Hierarchy for Applying Lists of Multiple Firewall Filters | **1343**
- Filter Input Lists and Output Lists for Router or Switch Interfaces | **1343**
- Types of Filters Supported in Lists | **1344**
- Restrictions on Applying Filter Lists for MPLS or Layer 2 CCC Traffic | **1344**

Statement Hierarchy for Applying Lists of Multiple Firewall Filters

To apply a single filter to the input or output direction of a router (or switch) *logical interface*, you include the `input` *`filter-name`* or `output` *`filter-name`* statement under the `filter` stanza for a protocol family.

To apply a list of multiple filters to the input or output direction of a router (or switch) logical interface, include the `input-list [` *`filter-names`* `]` or `output-list [` *`filter-names`* `]` statement under the `filter` stanza for a protocol family:

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

You can include the interface configuration at one of the following hierarchy levels:

- `[edit]`
- `[edit logical-systems` *`logical-system-name`*`]`

NOTE: (PTX10003) The router does not support `output-list` filter binding on the loopback address (lo0) or management interface.

Filter Input Lists and Output Lists for Router or Switch Interfaces

When applying a list of firewall filters as a list, the following limitations apply:

- You can specify up to 16 firewall filters for a filter input list.
- You can specify up to 16 firewall filters for a filter output list.

Types of Filters Supported in Lists

Lists of multiple firewall filters applied to a router (or switch) interface support standard stateless firewall filters only. You cannot apply lists containing service filters or simple filters to a router (or switch) interface.

Restrictions on Applying Filter Lists for MPLS or Layer 2 CCC Traffic

NOTE: These restrictions do not apply to the PTX10003 router. The router only supports applying filter lists on IPv4 (`inet`) or IPv6 (`inet6`) traffic.

When applying firewall filters that evaluate MPLS traffic (`family mpls`) or Layer 2 circuit cross-connection traffic (`family ccc`), you can use the `input-list [ filter-names ]` and `output-list [ filter-names ]` statements for all interfaces except the following:

- Management and internal Ethernet (`fxp`) interfaces
- Loopback (`lo0`) interfaces
- USB modem (`umd`) interfaces

RELATED DOCUMENTATION

Understanding Multiple Firewall Filters Applied as a List | 1338

Example: Applying Lists of Multiple Firewall Filters | 1344

Example: Applying Lists of Multiple Firewall Filters

IN THIS SECTION

- Requirements | 1345
- Overview | 1345
- Configuration | 1346
- Verification | 1351

This example shows how to apply lists of multiple firewall filters.

Requirements

Before you begin, be sure that you have:

- Installed your router or switch, and supported PIC, DPC, or MPC and performed the initial router or switch configuration.
- Configured basic Ethernet in the topology.
- Configured a logical interface to run the IP version 4 (IPv4) protocol (`family inet`) and configured the logical interface with an interface address. This example uses logical interface `ge-1/3/0.0` configured with the IP address 172.16.1.2/30.

  NOTE: For completeness, the configuration section of this example includes setting an IP address for logical interface `ge-1/3/0.0`.

- Verified that traffic is flowing in the topology and that ingress and egress IPv4 traffic is flowing through logical interface `ge-1/3/0.0`.
- Verified that you have access to the remote host that is connected to this router's or switch's logical interface `ge-1/3/0.0`.

NOTE: Physical interface policers/filters are not supported for list filters.

Overview

IN THIS SECTION

- Topology | **1346**

In this example, you configure three IPv4 firewall filters and apply each filter directly to the same logical interface by using a list.

Topology

This example applies the following firewall filters as a *list of input filters* at logical interface `ge-1/3/0.0`. Each filter contains a single term that evaluates IPv4 packets and accepts packets based on the value of the `destination port` field in the TCP header:

- Filter `filter_FTP` matches on the FTP port number (`21`).
- Filter `filter_SSH` matches on the SSH port number (`22`).
- Filter `filter_Telnet` matches on the Telnet port number (`23`).

If an inbound packet does not match any of the filters in the input list, the packet is discarded.

> **NOTE**: The Junos OS uses filters in a list in the order in which the filter names appear in the list. In this simple example, the order is irrelevant because all of the filters specify the same action.

Any of the filters can be applied to other interfaces, either alone (using the `input` or `output` statement) or in combination with other filters (using the `input-list` or `output-list` statement). The objective is to configure multiple “minimalist” firewall filters that you can reuse in interface-specific filter lists.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1347**
- Configure Multiple IPv4 Firewall Filters | **1347**
- Apply the Filters to a Logical Interface as an Input List and an Output List | **1348**
- Confirm and Commit Your Candidate Configuration | **1349**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the [edit] hierarchy level.

```
set firewall family inet filter filter_FTP term 0 from protocol tcp
set firewall family inet filter filter_FTP term 0 from destination-port 21
set firewall family inet filter filter_FTP term 0 then count pkts_FTP
set firewall family inet filter filter_FTP term 0 then accept
set firewall family inet filter filter_SSH term 0 from protocol tcp
set firewall family inet filter filter_SSH term 0 from destination-port 22
set firewall family inet filter filter_SSH term 0 then count pkts_SSH
set firewall family inet filter filter_SSH term 0 then accept
set firewall family inet filter filter_Telnet term 0 from protocol tcp
set firewall family inet filter filter_Telnet term 0 from destination-port 23
set firewall family inet filter filter_Telnet term 0 then count pkts_Telnet
set firewall family inet filter filter_Telnet term 0 then accept
set firewall family inet filter filter_discard term 1 then count pkts_discarded
set firewall family inet filter filter_discard term 1 then discard
set interfaces ge-1/3/0 unit 0 family inet address 172.16.1.2/30
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_FTP
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_SSH
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_Telnet
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_discard
```

Configure Multiple IPv4 Firewall Filters

Step-by-Step Procedure

To configure the IPv4 firewall filters:

1. Navigate the CLI to the hierarchy level at which you configure IPv4 firewall filters.

```
[edit]
user@host# edit firewall family inet
```

2. Configure the first firewall filter to count and accept packets for port 21.

```
[edit firewall family inet]
user@host# set filter filter_FTP term 0 from protocol tcp
```

```
user@host# set filter filter_FTP term 0 from destination-port 21
user@host# set filter filter_FTP term 0 then count pkts_FTP
user@host# set filter filter_FTP term 0 then accept
```

3. Configure the second firewall filter to count and accept packets for port 22.

```
[edit firewall family inet]
user@host# set filter filter_SSH term 0 from protocol tcp
user@host# set filter filter_SSH term 0 from destination-port 22
user@host# set filter filter_SSH term 0 then count pkt_SSH
user@host# set filter filter_SSH term 0 then accept
```

4. Configure the third firewall filter to count and accept packets from port 23.

```
[edit firewall family inet]
user@host# set filter filter_Telnet term 0 from protocol tcp
user@host# set filter filter_Telnet term 0 from destination-port 23
user@host# set filter filter_Telnet term 0 then count pkts_Telnet
user@host# set filter filter_Telnet term 0 then accept
```

5. Configure the last firewall filter to count the discarded packets.

```
[edit firewall family inet]
user@host# set filter filter_discard term 1 then count pkts_discarded
user@host# set filter filter_discard term 1 then discard
```

Apply the Filters to a Logical Interface as an Input List and an Output List

Step-by-Step Procedure

To apply the six IPv4 firewall filters as a list of input filters and a list of output filters:

1. Navigate the CLI to the hierarchy level at which you apply IPv4 firewall filters to logical interface `ge-1/3/0.0`.

```
[edit]
user@host# edit interfaces ge-1/3/0 unit 0 family inet
```

2. Configure the IPv4 protocol family for the logical interface.

```
[edit interfaces ge-1/3/0 unit 0 family inet]
user@host# set address 172.16.1.2/30
```

3. Apply the filters as a list of input filters.

```
[edit interfaces ge-1/3/0 unit 0 family inet]
user@host# set filter input-list [ filter_FTP filter_SSH filter_Telnet filter_discard ]
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filters by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter_FTP {
        term 0 {
            from {
                protocol tcp;
                destination-port 21;
            }
            then {
                count pkts_FTP;
                accept;
            }
        }
    }
    filter filter_SSH {
        term 0 {
            from {
                protocol tcp;
                destination-port 22;
```

```
            }
            then {
                count pkts_SSH;
                accept;
            }
        }
    }
    filter filter_Telnet {
        term 0 {
            from {
                protocol tcp;
                destination-port 23;
            }
            then {
                count pkts_Telnet;
                accept;
            }
        }
    }
    filter filter_discard {
        term 1 {
            then {
                count pkts_discarded;
                discard;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/0 {
    unit 0 {
        family inet {
            filter {
                input-list [ filter_FTP filter_SSH filter_Telnet filter_discard ];
            }
            address 172.16.1.2/30;
```

```
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

IN THIS SECTION

- Verifying That Inbound Packets Are Accepted Only If Destined for the FTP, SSH or Telnet Port | **1351**

Confirm that the configuration is working properly.

Verifying That Inbound Packets Are Accepted Only If Destined for the FTP, SSH or Telnet Port

Purpose

Verify that all three filters are active for the logical interface.

Action

To verify that input packets are accepted according to the three filters:

1. From the remote host that is connected to this router's (or switch's) logical interface `ge-1/3/0.0`, send a packet with destination port number 21 in the header. The packet should be accepted.
2. From the remote host that is connected to this router's (or switch's) logical interface `ge-1/3/0.0`, send a packet with destination port number 22 in the header. The packet should be accepted.
3. From the remote host that is connected to this router's (or switch's) logical interface `ge-1/3/0.0`, send a packet with destination port number 23 in the header. The packet should be accepted.
4. From the remote host that is connected to this router's (or switch's) logical interface `ge-1/3/0.0`, send a packet with a destination port number *other than* 21, 22, or 23. The packet should be discarded.

5. To display counter information for the list of filters applied to the input at `ge-1/3/0.0` enter the `show firewall filter ge-1/3/0.0-inet-i` operational mode command. The command output displays the number of bytes and packets that match filter terms associated with the following counters:

   - `pkts_FTP-ge-1/3/0.0-inet-i`
   - `pkts_SSH-ge-1/3/0.0-inet-i`
   - `pkts_Telnet-ge-1/3/0.0-inet-i`
   - `pkts_discard-ge-1/3/0.0-inet-i`

RELATED DOCUMENTATION

Understanding Multiple Firewall Filters Applied as a List | 1338

Guidelines for Applying Multiple Firewall Filters as a List | 1342

Example: Nesting References to Multiple Firewall Filters

IN THIS SECTION

- Requirements | 1352
- Overview | 1353
- Configuration | 1353
- Verification | 1357

This example shows how to configure nested references to multiple firewall filters.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1353

In this example, you configure a firewall filter for a match condition and action combination that can be shared among multiple firewall filters. You then configure two firewall filters that reference the first firewall filter. Later, if the common filtering criteria needs to be changed, you would modify only the one shared firewall filter configuration.

Topology

The `common_filter` firewall filter discards packets that have a UDP source or destination port field number of `69`. Both of the two additional firewall filters, `filter1` and `filter2`, reference the `common_filter`.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1353
- Configure the Nested Firewall Filters | 1354
- Apply Both Nested Firewall Filters to Interfaces | 1355
- Confirm and Commit Your Candidate Configuration | 1355

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter common_filter term common_term from protocol udp
set firewall family inet filter common_filter term common_term from port tftp
```

```
set firewall family inet filter common_filter term common_term then discard
set firewall family inet filter filter1 term term1 filter common_filter
set firewall family inet filter filter1 term term2 from address 192.168.0.0/16
set firewall family inet filter filter1 term term2 then reject
set firewall family inet filter filter2 term term1 filter common_filter
set firewall family inet filter filter2 term term2 from protocol udp
set firewall family inet filter filter2 term term2 from port bootps
set firewall family inet filter filter2 term term2 then accept
set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
set interfaces ge-0/0/0 unit 0 family inet filter input filter1
set interfaces ge-0/0/3 unit 0 family inet address 10.1.3.1/24
set interfaces ge-0/0/3 unit 0 family inet filter input filter2
```

Configure the Nested Firewall Filters

Step-by-Step Procedure

To configure two nested firewall filters that share a common filter:

1. Navigate the CLI to the hierarchy level at which you configure IPv4 firewall filters.

```
[edit]
user@host# edit firewall family inet
```

2. Configure the common filter that will be referenced by multiple other filters.

```
[edit firewall family inet]
user@host# set filter common_filter term common_term from protocol udp
user@host# set filter common_filter term common_term from port tftp
user@host# set filter common_filter term common_term then discard
```

3. Configure a filter that references the common filter.

```
[edit firewall family inet]
user@host# set filter filter1 term term1 filter common_filter
user@host# set filter filter1 term term2 from address 192.168.0.0/16
user@host# set filter filter1 term term2 then reject
```

4. Configure a second filter that references the common filter.

```
[edit firewall family inet]
user@host# set filter filter2 term term1 filter common_filter
user@host# set filter filter2 term term2 from protocol udp
user@host# set filter filter2 term term2 from port bootps
user@host# set filter filter2 term term2 then accept
```

Apply Both Nested Firewall Filters to Interfaces

Step-by-Step Procedure

To apply both nested firewall filters to logical interfaces:

1. Apply the first nested filter to a logical interface input.

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
user@host# set interfaces ge-0/0/0 unit 0 family inet filter input filter1
```

2. Apply the second nested filter to a logical interface input.

```
[edit]
user@host# set interfaces ge-0/0/3 unit 0 family inet address 10.1.3.1/24
user@host# set interfaces ge-0/0/3 unit 0 family inet filter input filter2
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
```

```
    filter common_filter {
        term common_term {
            from {
                protocol udp;
                port tftp;
            }
            then {
                discard;
            }
        }
    }
    filter filter1 {
        term term1 {
            filter common_filter;
        }
        term term2 {
            from {
                address 192.168/16;
            }
            then {
                reject;
            }
        }
    }
    filter filter2 {
        term term1 {
            filter common_filter;
        }
        term term2 {
            from {
                protocol udp;
                port bootps;
            }
            then {
                accept;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            filter {
                input filter1;
            }
            address 10.1.0.1/24;
        }
    }
}
ge-0/0/3 {
    unit 0 {
        family inet {
            filter {
                input filter2;
            }
            address 10.1.3.1/24;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show firewall filter filter1` and `show firewall filter filter2` operational mode commands.

RELATED DOCUMENTATION

Understanding Multiple Firewall Filters in a Nested Configuration | 1334

Guidelines for Nesting References to Multiple Firewall Filters | 1336

Example: Filtering Packets Received on an Interface Set

IN THIS SECTION

- Requirements | 1358
- Overview | 1358
- Configuration | 1359
- Verification | 1367

This example shows how to configure a standard stateless firewall filter to match packets tagged for a particular interface set.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1358

In this example, you apply a stateless firewall filter to the input of the router or switch loopback interface. The firewall filter includes a term that matches packets tagged for a particular interface set.

Topology

You create the firewall filter `L2_filter` to apply rate limits to the protocol-independent traffic received on the following interfaces:

- `fe-0/0/0.0`

- fe-1/0/0.0
- fe-1/1/0.0

NOTE: The interface type in this topic is just an example. The fe- interface type is not supported by EX Series switches.

First, for protocol-independent traffic received on fe-0/0/0.0, the firewall filter term t1 applies policer p1.

For protocol-independent traffic received on any other Fast Ethernet interfaces, firewall filter term t2 applies policer p2. To define an interface set that consists of all Fast Ethernet interfaces, you include the interface-set *interface-set-name interface-name* statement at the [edit firewall] hierarchy level. To define a packet-matching criteria based on the interface on which a packet arrives to a specified interface set, you configure a term that uses the interface-set firewall filter match condition.

Finally, for any other protocol-independent traffic, firewall filter term t3 applies policer p3.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1360**
- Configuring the Interfaces for Which the Stateless Firewall Filter Terms Take Rate-Limiting Actions | **1360**
- Configuring the Stateless Firewall Filter That Rate-Limits Protocol-Independent Traffic Based on the Interfaces on Which Packets Arrive | **1362**
- Applying the Stateless Firewall Filter to the Routing Engine Input Interface | **1365**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces fe-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces fe-1/0/0 unit 0 family inet address 10.2.2.1/30
set interfaces fe-1/1/0 unit 0 family inet address 10.4.4.1/30
set firewall policer p1 if-exceeding bandwidth-limit 5m
set firewall policer p1 if-exceeding burst-size-limit 10m
set firewall policer p1 then discard
set firewall policer p2 if-exceeding bandwidth-limit 40m
set firewall policer p2 if-exceeding burst-size-limit 100m
set firewall policer p2 then discard
set firewall policer p3 if-exceeding bandwidth-limit 600m
set firewall policer p3 if-exceeding burst-size-limit 1g
set firewall policer p3 then discard
set firewall interface-set ifset fe-*
set firewall family any filter L2_filter term t1 from interface fe-0/0/0.0
set firewall family any filter L2_filter term t1 then count c1
set firewall family any filter L2_filter term t1 then policer p1
set firewall family any filter L2_filter term t2 from interface-set ifset
set firewall family any filter L2_filter term t2 then count c2
set firewall family any filter L2_filter term t2 then policer p2
set firewall family any filter L2_filter term t3 then count c3
set firewall family any filter L2_filter term t3 then policer p3
set interfaces lo0 unit 0 family inet address 172.16.1.157/30
set interfaces lo0 unit 0 family inet address 172.16.1.157/30
set interfaces lo0 unit 0 filter input L2_filter
```

Configuring the Interfaces for Which the Stateless Firewall Filter Terms Take Rate-Limiting Actions

Step-by-Step Procedure

To configure the interfaces for which the stateless firewall filter terms take rate-limiting actions:

1. Configure the logical interface whose input traffic will be matched by the first term of the firewall filter.

```
[edit]
user@host# set interfaces fe-0/0/0 unit 0 family inet address 10.1.1.1/30
```

2. Configure the logical interfaces whose input traffic will be matched by the second term of the firewall filter.

```
[edit ]
user@host# set interfaces fe-1/0/0 unit 0 family inet address 10.2.2.1/30
user@host# set interfaces fe-1/1/0 unit 0 family inet address 10.4.4.1/30
```

3. If you are done configuring the device, commit the configuration.

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

Results

Confirm the configuration of the router (or switch) transit interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show 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;
        }
    }
```

```
}
fe-1/1/0 {
    unit 0 {
        family inet {
            address 10.4.4.1/30;
        }
    }
}
```

Configuring the Stateless Firewall Filter That Rate-Limits Protocol-Independent Traffic Based on the Interfaces on Which Packets Arrive

Step-by-Step Procedure

To configure the standard stateless firewall `L2_filter` that uses policers (`p1`, `p2`, and `p3`) to rate-limit protocol-independent traffic based on the interfaces on which the packets arrive:

1. Configure the firewall statements.

```
[edit]
user@host# edit firewall
```

2. Configure the policer `p1` to discard traffic that exceeds a traffic rate of `5m` bps or a burst size of `10m` bytes.

```
[edit firewall]
user@host# set policer p1 if-exceeding bandwidth-limit 5m
user@host# set policer p1 if-exceeding burst-size-limit 10m
user@host# set policer p1 then discard
```

3. Configure the policer `p2` to discard traffic that exceeds a traffic rate of `40m` bps or a burst size of `100m` bytes .

```
[edit firewall]
user@host# set policer p2 if-exceeding bandwidth-limit 40m
user@host# set policer p2 if-exceeding burst-size-limit 100m
user@host# set policer p2 then discard
```

4. Configure the policer p3 to discard traffic that exceeds a traffic rate of 600m bps or a burst size of 1g bytes.

```
[edit firewall]
user@host# set policer p3 if-exceeding bandwidth-limit 600m
user@host# set policer p3 if-exceeding burst-size-limit 1g
user@host# set policer p3 then discard
```

5. Define the interface set ifset to be the group of all Fast Ethernet interfaces on the router.

```
[edit firewall]
user@host# set interface-set ifset fe-*
```

6. Create the stateless firewall filter L2_filter.

```
[edit firewall]
user@host# edit family any filter L2_filter
```

7. Configure filter term t1 to match IPv4, IPv6, or MPLS packets received on interface fe-0/0/0.0 and use policer p1 to rate-limit that traffic.

```
[edit firewall family any filter L2_filter]
user@host# set term t1 from interface fe-0/0/0.0
user@host# set term t1 then count c1
user@host# set term t1 then policer p1
```

8. Configure filter term t2 to match packets received on interface-set ifset and use policer p2 to rate-limit that traffic.

```
[edit firewall family any filter L2_filter]
user@host# set term t2 from interface-set ifset
user@host# set term t2 then count c2
user@host# set term t2 then policer p2
```

9. Configure filter term `t3` to use policer `p3` to rate-limit all other traffic.

```
[edit firewall family any filter L2_filter]
user@host# set term t3 then count c3
user@host# set term t3 then policer p3
```

10. If you are done configuring the device, commit the configuration.

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

Results

Confirm the configuration of the stateless firewall filter and the policers referenced as firewall filter actions by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family any {
    filter L2_filter {
        term t1 {
            from {
                interface fe-0/0/0.0;
            }
            then {
                policer p1;
                count c1;
            }
        }
        term t2 {
            from {
                interface-set ifset;
            }
            then {
                policer p2;
                count c2;
            }
        }
```

```
        term t3 {
            then {
                policer p3;
                count c3;
            }
        }
    }
}
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-*;
}
```

Applying the Stateless Firewall Filter to the Routing Engine Input Interface

Step-by-Step Procedure

To apply the stateless firewall filter to the Routing Engine input interface:

1. Apply the stateless firewall filter to the Routing Engine interface in the input direction.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet address 172.16.1.157/30
user@host# set interfaces lo0 unit 0 filter input L2_filter
```

2. If you are done configuring the device, commit the configuration.

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

Results

Confirm the application of the firewall filter to the Routing Engine input interface by entering the `show interfaces` command again. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
user@host# show interfaces
fe-0/0/0 {
    ...
}
fe-1/0/0 {
    ...
}
fe-1/1/0 {
    ...
}
lo0 {
    unit 0 {
        filter {
            input L2_filter;
        }
        family inet {
            address 172.16.1.157/30;
        }
    }
}
```

Verification

To confirm that the configuration is working properly, use the `show firewall filter L2_filter` operational mode command to monitor traffic statistics about the firewall filter and three counters.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Filtering Packets Received on an Interface Set Overview | 1374

CHAPTER 20

Attaching a Single Firewall Filter to Multiple Interfaces

IN THIS CHAPTER

- Interface-Specific Firewall Filter Instances Overview | 1368
- Interface-Specific Firewall Filter Instances Overview | 1371
- Filtering Packets Received on a Set of Interface Groups Overview | 1373
- Filtering Packets Received on an Interface Set Overview | 1374
- Example: Configuring Interface-Specific Firewall Filter Counters | 1375
- Example: Configuring a Stateless Firewall Filter on an Interface Group | 1382

Interface-Specific Firewall Filter Instances Overview

IN THIS SECTION

- Instantiation of Interface-Specific Firewall Filters | 1368
- Interface-Specific Names for Firewall Filter Instances | 1369
- Interface-Specific Firewall Filter Counters | 1370
- Interface-Specific Firewall Filter Policers | 1370

Instantiation of Interface-Specific Firewall Filters

On T Series, M120, M320, MX Series routers, and EX Series switches, you can enable the Junos OS to automatically create an interface-specific instance of a firewall filter for each interface to which you apply the filter. If you enable interface-specific instantiation of a firewall filter and then apply that filter to multiple interfaces, any **count** actions or **policer** actions configured in the filter terms act on the traffic

stream entering or exiting each individual interface, regardless of the sum of traffic on the multiple interfaces.

You can enable this option per firewall filter by including the `interface-specific` statement in the filter configuration.

> **NOTE**: On T Series, M120, M320, MX Series routers, and EX Series switches, interfaces are distributed among multiple packet-forwarding components.

Interface-specific firewall filtering is not supported on M Series routers other than the M120 and M320 routers. If you apply a firewall filter to multiple interfaces on an M Series router other than the M120 or M320 routers, the filter acts on the sum of traffic entering or exiting those interfaces.

Interface-specific firewall filtering is supported for standard stateless firewall filters and for service filters. Interface-specific instances are not supported for simple filters.

Interface-Specific Names for Firewall Filter Instances

When the Junos OS creates a separate instance of a firewall filter for a logical interface, the instance is associate with an interface-specific name. The system-generated name of a firewall filter instance consists of the name of the configured filter followed by a hyphen ('**-**'), the full interface name, and either '**-i**' for an input filter instance or '**-o**' for an output filter instance.

- **Input filter instance name**—For example, if you apply the interface-specific firewall filter **filter_s_tcp** to the input at logical interface **at-1/1/1.0**, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

  ```
  filter_s_tcp-at-1/1/1.0-i
  ```

- **Output filter instance name**—For example, if you apply the interface-specific firewall filter **filter_s_tcp** to the output at logical interface **ge-2/2/2.2**, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

  ```
  count_s_tcp-ge-2/2/2.2-o
  ```

You can use the interface-specific name of a filter instance when you enter a Junos OS operational mode command that specifies a stateless firewall filter name.

TIP: When you configure a firewall filter with interface-specific instances enabled, we recommend you limit the filter name to *52 bytes* in length. This is because firewall filter names are restricted to *64 bytes* in length. If a system-generated filter instance name exceeds this maximum length, the policy framework software might reject the instance name.

Interface-Specific Firewall Filter Counters

Instantiation of interface-specific firewall filters causes the Packet Forwarding Engine to maintain any counters for the firewall filter separately for each interface. You specify interface-specific counters per firewall filter term by specifying the **count** *counter-name* non-terminating action.

The system-generated name of an interface-specific firewall filter counter consists of the name of the configured counter followed by a hyphen ('**-**'), the full interface name, and either '**-i**' for an input filter instance or '**-o**' for an output filter instance.

- **Interface-specific input filter counter name**—For example, suppose you configure the filter counter **count_tcp** for an interface-specific firewall filter. If the filter is applied to the input at logical interface **at-1/1/1.0**, the Junos OS creates the following system-generated counter name:

```
count_tcp-at-1/1/1.0-i
```

- **Interface-specific output filter counter name**—For example, suppose you configure the filter counter **count_udp** for an interface-specific firewall filter. If the filter is applied to the output at logical interface **ge-2/2/2.2**, the Junos OS creates the following system-generated counter name:

```
count_udp-ge-2/2/2.2-o
```

Interface-Specific Firewall Filter Policers

Instantiation of interface-specific firewall filters not only creates separate instances of any firewall filter counters but also creates separate instances of any policer actions. Any policers applied through an action specified in the firewall filter configuration are applied separately to each interface in the interface group. You specify interface-specific policers per firewall filter term by specifying the **policer** *policer-name* non-terminating action.

RELATED DOCUMENTATION

Example: Configuring Interface-Specific Firewall Filter Counters | 1375

Interface-Specific Firewall Filter Instances Overview

IN THIS SECTION

- Instantiation of Interface-Specific Firewall Filters | 1371
- Interface-Specific Names for Firewall Filter Instances | 1372
- Interface-Specific Firewall Filter Counters | 1372
- Interface-Specific Firewall Filter Policers | 1373

Instantiation of Interface-Specific Firewall Filters

On T Series, M120, M320, and MX Series routers, you can enable the Junos OS to automatically create an interface-specific instance of a *firewall filter* for each interface to which you apply the filter. If you enable interface-specific instantiation of a firewall filter and then apply that filter to multiple interfaces, any `count` actions or `policer` actions configured in the filter terms act on the traffic stream entering or exiting each individual interface, regardless of the sum of traffic on the multiple interfaces.

You can enable this option per firewall filter by including the `interface-specific` statement in the filter configuration.

NOTE: On T Series, M120, M320, and MX Series routers, interfaces are distributed among multiple packet-forwarding components.

Interface-specific firewall filtering is not supported on M Series routers other than the M120 and M320 routers. If you apply a firewall filter to multiple interfaces on an M Series router other than the M120 or M320 routers, the filter acts on the sum of traffic entering or exiting those interfaces.

Interface-specific firewall filtering is supported for standard stateless firewall filters and for service filters. Interface-specific instances are not supported for simple filters.

NOTE: A firewall filter cannot be both interface-specific and interface-shared.

Interface-Specific Names for Firewall Filter Instances

When the Junos OS creates a separate instance of a firewall filter for a *logical interface*, the instance is associate with an interface-specific name. The system-generated name of a firewall filter instance consists of the name of the configured filter followed by a hyphen ('-'), the full interface name, and either '`-i`' for an input filter instance or '`-o`' for an output filter instance.

- **Input filter instance name**—For example, if you apply the interface-specific firewall filter `filter_s_tcp` to the input at logical interface `at-1/1/1.0`, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

```
filter_s_tcp-at-1/1/1.0-i
```

- **Output filter instance name**—For example, if you apply the interface-specific firewall filter `filter_s_tcp` to the output at logical interface `so-2/2/2.2`, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

```
count_s_tcp-so-2/2/2.2-o
```

You can use the interface-specific name of a filter instance when you enter a Junos OS *operational mode command* that specifies a stateless firewall filter name.

TIP: When you configure a firewall filter with interface-specific instances enabled, we recommend you limit the filter name to *52 bytes* in length. This is because firewall filter names are restricted to *64 bytes* in length. If a system-generated filter instance name exceeds this maximum length, the policy framework software might reject the instance name.

Interface-Specific Firewall Filter Counters

Instantiation of interface-specific firewall filters causes the Packet Forwarding Engine to maintain any counters for the firewall filter separately for each interface. You specify interface-specific counters per firewall filter term by specifying the `count` *`counter-name`* non-terminating action.

The system-generated name of an interface-specific firewall filter counter consists of the name of the configured counter followed by a hyphen ('-'), the full interface name, and either '-i' for an input filter instance or '-o' for an output filter instance.

- **Interface-specific input filter counter name**—For example, suppose you configure the filter counter count_tcp for an interface-specific firewall filter. If the filter is applied to the input at logical interface at-1/1/1.0, the Junos OS creates the following system-generated counter name:

```
count_tcp-at-1/1/1.0-i
```

- **Interface-specific output filter counter name**—For example, suppose you configure the filter counter count_udp for an interface-specific firewall filter. If the filter is applied to the output at logical interface so-2/2/2.2, the Junos OS creates the following system-generated counter name:

```
count_udp-so-2/2/2.2-o
```

Interface-Specific Firewall Filter Policers

Instantiation of interface-specific firewall filters not only creates separate instances of any firewall filter counters but also creates separate instances of any policer actions. Any policers applied through an action specified in the firewall filter configuration are applied separately to each interface in the interface group. You specify interface-specific policers per firewall filter term by specifying the policer *policer-name* non-terminating action.

RELATED DOCUMENTATION

Example: Configuring Interface-Specific Firewall Filter Counters | 1375

Filtering Packets Received on a Set of Interface Groups Overview

You can configure a *firewall filter* term that matches packets tagged for a specified *interface group* or set of interface groups. An interface group consists of one or more logical interfaces with the same group number. Packets received on an interface in an interface group are tagged as being part of that group.

NOTE: EX9200 switches do not support *interface groups*. Use the interface-set configuration command as a workaround.

For standard stateless firewall filters, you can filter packets received on an interface group for IPv4, IPv6, virtual private LAN service (VPLS), Layer 2 circuit cross-connection (CCC), and Layer 2 bridging traffic. For service filters, you can filter packets received on an interface group for either IPv4 or IPv6 traffic.

NOTE: You can also configure a firewall filter term that matches on packets tagged for a specified *interface set*. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 1374.

RELATED DOCUMENTATION

Example: Configuring a Stateless Firewall Filter on an Interface Group | **1382**

Filtering Packets Received on an Interface Set Overview

You can configure a standard stateless *firewall filter* term that matches packets tagged for a specified *interface set*. An interface set groups two or more physical or logical interfaces into a single interface-set name. You can filter packets received on an interface set for protocol-independent, IPv4, IPv6, MPLS, VPLS, or bridging traffic.

NOTE: You can also configure a standard stateless firewall filter term or a service filter term that matches on packets tagged for a specified *interface group*. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.

RELATED DOCUMENTATION

Example: Configuring a Rate-Limiting Filter Based on Destination Class | **1208**

Example: Filtering Packets Received on an Interface Set | **1358**

Example: Configuring Interface-Specific Firewall Filter Counters

IN THIS SECTION

- Requirements | 1375
- Overview | 1375
- Configuration | 1376
- Verification | 1380

This example shows how to configure and apply an interface-specific standard stateless firewall filter.

Requirements

Interface-specific stateless firewall filters are supported on T Series, M120, M320, and MX Series routers only.

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1375

In this example, you create an interface-specific stateless firewall filter that counts and accepts packets with source or destination addresses in a specified prefix and the IP protocol type field set to a specific value.

Topology

You configure the interface-specific stateless firewall filter `filter_s_tcp` to count and accept packets with IP source or destination addresses in the `10.0.0.0/12` prefix and the IP protocol type field set to `tcp` (or the numeric value `6`).

The name of the firewall filter counter is `count_s_tcp`.

You apply the firewall filter to multiple logical interfaces:

- `at-1/1/1.0` input
- `so-2/2/2.2` output

Applying the filter to these two interfaces results in two instances of the filter: `filter_s_tcp-at-1/1/1.0-i` and `filter_s_tcp-so-2/2/2.2-o`, respectively.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1376**
- Configure the Interface-Specific Firewall Filter | **1377**
- Apply the Interface-Specific Firewall Filter to Multiple Interfaces | **1377**
- Confirm Your Candidate Configuration | **1378**
- Clear the Counters and Commit Your Candidate Configuration | **1379**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet filter filter_s_tcp interface-specific
set firewall family inet filter filter_s_tcp term 1 from address 10.0.0.0/12
set firewall family inet filter filter_s_tcp term 1 from protocol tcp
set firewall family inet filter filter_s_tcp term 1 then count count_s_tcp
set firewall family inet filter filter_s_tcp term 1 then accept
set interfaces at-1/1/1 unit 0 family inet filter input filter_s_tcp
set interfaces so-2/2/2 unit 2 family inet filter filter_s_tcp
```

Configure the Interface-Specific Firewall Filter

Step-by-Step Procedure

To configure the interface-specific firewall filter:

1. Create the IPv4 firewall filter `filter_s_tcp`.

```
[edit]
user@host# edit firewall family inet filter filter_s_tcp
```

2. Enable interface-specific instances of the filter.

```
[edit firewall family inet filter filter_s_tcp]
user@host# set interface-specific
```

3. Configure the match conditions for the term.

```
[edit firewall family inet filter filter_s_tcp]
user@host# set term 1 from address 10.0.0.0/12
user@host# set term 1 from protocol tcp
```

4. Configure the actions for the term.

```
[edit firewall family inet filter filter_s_tcp]
user@host# set term 1 then count count_s_tcp
user@host# set term 1 then accept
```

Apply the Interface-Specific Firewall Filter to Multiple Interfaces

Step-by-Step Procedure

To apply the filter `filter_s_tcp` to logical interfaces `at-1/1/1.0` and `so-2/2/2.2`:

1. Apply the interface-specific filter to packets received on logical interface `at-1/1/1.0`.

```
[edit]
user@host# set interfaces at-1/1/1 unit 0 family inet filter input filter_s_tcp
```

2. Apply the interface-specific filter to packets transmitted from logical interface `so-2/2/2.2`.

```
[edit]
user@host# set interfaces so-2/2/2 unit 2 family inet filter filter_s_tcp
```

Confirm Your Candidate Configuration

Step-by-Step Procedure

To confirm your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter_s_tcp {
        interface-specific;
        term 1 {
            from {
                address {
                    10.0.0.0/12;
                }
                protocol tcp;
            }
            then {
                count count_s_tcp;
                accept;
            }
        }
    }
}
```

2. Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
at-1/1/1 {
    unit 0
        family inet {
            filter {
                input filter_s_tcp;
            }
        }
    ]
}
so-2/2/2 {
    unit 2
        family inet {
            filter {
                output filter_s_tcp;
            }
        }
    }
}
```

Clear the Counters and Commit Your Candidate Configuration

Step-by-Step Procedure

To clear the counters and commit your candidate configuration:

1. From operational command mode, use the `clear firewall all` command to clear the statistics for all firewall filters.

   To clear only the counters used in this example, include the interface-specific filter instance names:

```
[edit]
user@host> clear firewall filter filter_s_tcp-at-1/1/1.0-i
user@host> clear firewall filter filter_s_tcp-so-2/2/2.2-o
```

2. Commit your candidate configuration.

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

Verification

IN THIS SECTION

- Verifying That the Filter Is Applied to Each of the Multiple Interfaces | 1380
- Verifying That the Counters Are Collected Separately by Interface | 1381

Confirm that the configuration is working properly.

Verifying That the Filter Is Applied to Each of the Multiple Interfaces

Purpose

Verify that the filter is applied to each of the multiple interfaces.

Action

Run the `show interfaces` command with the `detail` or `extensive` output level.

1. Verify that the filter is applied to the input for `at-1/1/1.0`:

```
user@host> show interfaces at-1/1/1 detail
Physical interface: at-1/1/1, Enabled, Physical link is Up
  Interface index: 300, SNMP ifIndex: 194, Generation: 183

...
  Logical interface at-1/1/1.0 (Index 64) (SNMP ifIndex 204) (Generation 5)
    Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: ATM-SNAP
...
    Protocol inet, MTU: 4470, Generation: 13, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter_s_tcp-at-1/1/1.0-i,,,,,
```

2. Verify that the filter is applied to the output for `so-2/2/2.2`:

```
user@host> show interfaces so-2/2/2 detail
Physical interface: so-2/2/2, Enabled, Physical link is Up
  Interface index: 129, SNMP ifIndex: 502, Generation: 132

...
  Logical interface so-2/2/2.2 (Index 70) (SNMP ifIndex 536) (Generation 135)
    Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP

...
    Protocol inet, MTU: 4470, Generation: 146, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Output Filters: filter_s_tcp-so-2/2/2.2-o,,,,,
```

Verifying That the Counters Are Collected Separately by Interface

Purpose

Make sure that the `count_s_tcp` counters are collected separately for the two logical interfaces.

Action

Run the `show firewall` command.

```
user@host> show firewall filter filter_s_tcp
Filter: filter_s_tcp
Counters:
Name                              Bytes             Packets
count_s_tcp-at-1/1/1.0-i            420                   5
count_s_tcp-so-2/2/2.2-o           8888                 101
```

RELATED DOCUMENTATION

Interface-Specific Firewall Filter Instances Overview | 1371

Example: Configuring a Stateless Firewall Filter on an Interface Group

IN THIS SECTION

- Requirements | 1382
- Overview | 1382
- Configuration | 1383
- Verification | 1388

Firewall filters are essential for securing a network and simplifying network management. In Junos OS, you can configure a stateless firewall filters to control the transit of data packets through the system and to manipulate packets as necessary. Applying a stateless firewall filter to an interface group helps to filter packets transiting through each interface in the interface group. This example shows how to configure a standard stateless firewall filter to match packets tagged for a particular interface group.

Requirements

This example uses the following hardware and software components:

- Any two Juniper Networks routers or switches that are physically or logically connected to each other through interfaces belonging to a routing instance
- Junos OS Release 7.4 or later

Overview

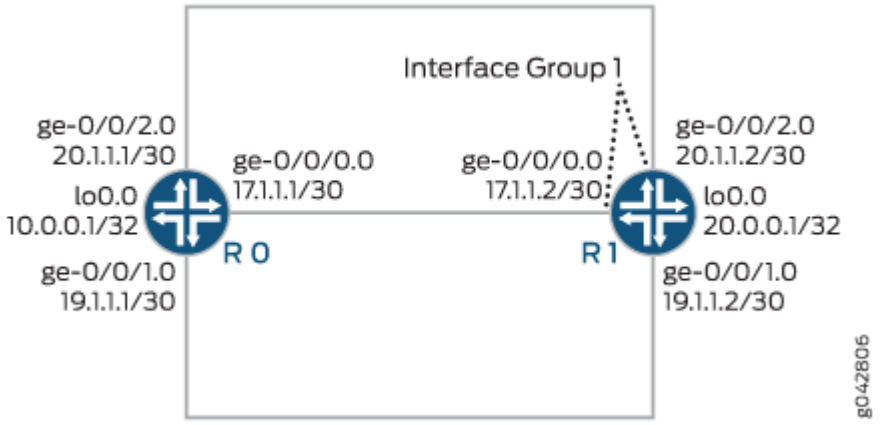
You can apply a stateless firewall filter to an interface group to apply it across all the interfaces in the interface group. This helps you to manage the packet filtering on various interfaces simultaneously.

In this example, you configure two router or switch interfaces to belong to the interface group. You also configure a stateless firewall filter with three terms. In term `term1`, the filter matches packets that have been tagged as received on that interface group and contain an ICMP protocol tag. The filter counts, logs, and rejects packets that match the conditions. In term `term2`, the filter matches packets that contain the ICMP protocol tag. The filter counts, logs, and accepts all packets that match the condition. In term `term3`, the filter counts all the transit packets.

By applying the firewall filter to the routing instance, you can simultaneously apply the filtering mechanism on all the interfaces in the interface group. For this to happen, all the interfaces in the interface group must belong to a single routing instance.

NOTE: When you apply a firewall filter to a loopback interface, the interface filters all the packets destined to the Routing Engine.

Figure 60: Configuring a Stateless Firewall Filter on an Interface Group



CLI Quick Configuration shows the configuration for all of the devices in Figure 60 on page 1383. The section Step-by-Step Procedure describes the steps on Device R1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1383**
- Configure and Apply the Stateless Firewall Filter on an Interface Group | **1384**
- Results | **1386**

CLI Quick Configuration

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

Device R0

```
set interfaces ge-0/0/0 unit 0 family inet address 172.16.17.1/30
set interfaces ge-0/0/1 unit 0 family inet address 172.16.19.1/30
set interfaces ge-0/0/2 unit 0 family inet address 20.1.1.1/30
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
```

Device R1

```
set firewall family inet filter filter_if_group term term1 from interface-group 1
set firewall family inet filter filter_if_group term term1 from protocol icmp
set firewall family inet filter filter_if_group term term1 then count if_group_counter1
set firewall family inet filter filter_if_group term term1 then log
set firewall family inet filter filter_if_group term term1 then reject
set firewall family inet filter filter_if_group term term2 from protocol icmp
set firewall family inet filter filter_if_group term term2 then count if_group_counter2
set firewall family inet filter filter_if_group term term2 then log
set firewall family inet filter filter_if_group term term2 then accept
set firewall family inet filter filter_if_group term term3 then count default
set interfaces ge-0/0/0 unit 0 family inet filter group 1
set interfaces ge-0/0/0 unit 0 family inet address 172.16.17.2/30
set interfaces ge-0/0/1 unit 0 family inet address 172.16.19.2/30
set interfaces ge-0/0/2 unit 0 family inet filter group 1
set interfaces ge-0/0/2 unit 0 family inet address 20.1.1.2/30
set interfaces lo0 unit 0 family inet address 20.0.0.1/32
set forwarding-options family inet filter input filter_if_group
```

Configure and Apply the Stateless Firewall Filter on an Interface Group

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the CLI User Guide.

To configure the stateless firewall filter `filter_if_group` on an interface group:

1. Create the stateless firewall filter `filter_if_group`.

```
[edit firewall]
user@R1# edit family inet filter filter_if_group
```

2. Configure the interfaces and assign two interfaces to interface group `1`.

```
[edit interfaces]
user@R1# set ge-0/0/0 unit 0 family inet filter group 1
user@R1# set ge-0/0/0 unit 0 family inet address 172.16.17.2/30
user@R1# set ge 0/0/1 unit 0 family inet address 172.16.19.2/30
user@R1# set ge-0/0/2 unit 0 family inet filter group 1
user@R1# set ge-0/0/2 unit 0 family inet address 20.1.1.2/30
user@R1# set lo0 unit 0 family inet address 20.0.0.1/32
```

3. Configure term `term1` to match packets received on interface group `1` and with the ICMP protocol.

```
[edit firewall]
user@R1# set family inet filter filter_if_group term term1 from interface-group 1
user@R1# set family inet filter filter_if_group term term1 from protocol icmp
```

4. Configure term `term1` to count, log, and reject all the matching packets.

```
[edit firewall]
user@R1# set family inet filter filter_if_group term term1 then count if_group_counter1
user@R1# set family inet filter filter_if_group term term1 then log
user@R1# set family inet filter filter_if_group term term1 then reject
```

5. Configure term `term2` to match packets with the ICMP protocol.

```
[edit firewall]
user@R1# set family inet filter filter_if_group term term2 from protocol icmp
```

6. Configure term `term2` to count, log, and accept all the matching packets.

```
[edit firewall]
user@R1# set family inet filter filter_if_group term term2 then count if_group_counter2
```

```
user@R1# set family inet filter filter_if_group term term2 then log
user@R1# set family inet filter filter_if_group term term2 then accept
```

7. Configure term `term3` to count all the transit packets.

```
[edit firewall]
user@R1# set family inet filter filter_if_group term term3 then count default
```

8. Apply the firewall filter to the router's (or switch's) interface group by applying it to the routing instance.

```
[edit]
user@R1# set forwarding-options family inet filter input filter_if_group
```

9. If you are done configuring the device, commit your candidate configuration.

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

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show firewall`, and `show forwarding-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@R1# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            filter {
                group 1;
            }
            address 172.16.17.2/30;
        }
    }
}
ge-0/0/1 {
```

```
    unit 0 {
        family inet {
            address 172.16.19.2/30;
        }
    }
}
ge-0/0/2 {
    unit 0 {
        family inet {
            filter {
                group 1;
            }
            address 20.1.1.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 20.0.0.1/32;
        }
    }
}
```

```
[edit]
user@R1# show firewall
family inet {
    filter filter_if_group {
        term term1 {
            from {
                interface-group 1;
                protocol icmp;
            }
            then {
                count if_group_counter1;
                log;
                reject;
            }
        }
        term term2 {
            from {
```

```
                protocol icmp;
            }
            then {
                count if_group_counter2;
                log;
                accept;
            }
        }
        term term3 {
            then count default;
        }
    }
}
```

```
[edit]
user@R1# show forwarding-options
family inet {
    filter {
        input filter_if_group;
    }
}
```

Verification

IN THIS SECTION

- Verifying the Configuration of the Interfaces | **1388**
- Verifying Stateless Firewall Filter Configuration | **1390**

Confirm that the configuration is working properly.

Verifying the Configuration of the Interfaces

Purpose

Verify that the interfaces are properly configured.

Action

To display the state of the interfaces, use the `show interfaces terse` operational mode command.

Device R0

```
user@R0> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.0              up    up   inet     172.16.17.1/30
                                  multiservice
ge-0/0/1                up    up
ge-0/0/1.0              up    up   inet     172.16.19.1/30
                                  multiservice
ge-0/0/2                up    up
ge-0/0/2.0              up    up   inet     20.1.1.1/30
                                  multiservice
lo0                     up    up
lo0.0                   up    up   inet     10.0.0.1            --> 0/0
```

Device R1

```
user@R1> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.0              up    up   inet     172.16.17.2/30
                                  multiservice
...
ge-0/0/1                up    up
ge-0/0/1.0              up    up   inet     172.16.19.2/30
                                  multiservice
ge-0/0/2                up    up
ge-0/0/2.0              up    up   inet     20.1.1.2/30
                                  multiservice
...
```

Meaning

All the interfaces on Devices R0 and R1 are physically connected and up. The interface group `1` on Device R1 consists of two interfaces, namely ge-0/0/0.0 and ge-0/0/2.0.

Verifying Stateless Firewall Filter Configuration

Purpose

Verify that the firewall filter match conditions are configured properly.

Action

- To display the firewall filter counters, enter the `show firewall filter filter_if_group` operational mode command.

```
user@R1> show firewall filter filter_if_group

Filter: filter_if_group
Counters:
Name                                                Bytes              Packets
default                                            192975                 3396
if_group_counter1                                    2520                   30
if_group_counter2                                    2604                   41
```

- To display the local log of packet headers for packets evaluated by the firewall filter, enter the `show firewall log` operational mode command.

```
user@R1> show firewall log
Log :
Time      Filter    Action Interface     Protocol        Src Addr
Dest Addr
22:27:33  pfe       A      lo0.0         ICMP            20.1.1.2
20.1.1.1
22:27:33  pfe       R      ge-0/0/2.0    ICMP            20.1.1.1
20.1.1.2
22:27:32  pfe       A      lo0.0         ICMP            20.1.1.2
20.1.1.1
22:27:32  pfe       R      ge-0/0/2.0    ICMP            20.1.1.1
20.1.1.2
22:27:31  pfe       A      lo0.0         ICMP            20.1.1.2
20.1.1.1
22:27:31  pfe       R      ge-0/0/2.0    ICMP            20.1.1.1
20.1.1.2
22:27:30  pfe       A      lo0.0         ICMP            20.1.1.2
20.1.1.1
```

```
22:27:30  pfe       R      ge-0/0/2.0    ICMP           20.1.1.1
20.1.1.2
22:27:29  pfe       A      lo0.0         ICMP           20.1.1.2
20.1.1.1
22:27:29  pfe       A      lo0.0         ICMP           20.1.1.2
20.1.1.1
22:27:29  pfe       R      ge-0/0/2.0    ICMP           20.1.1.1
20.1.1.2
22:27:21  pfe       A      ge-0/0/1.0    ICMP           172.16.19.1
172.16.19.2
22:27:20  pfe       A      ge-0/0/1.0    ICMP           172.16.19.1
172.16.19.2
22:27:19  pfe       A      ge-0/0/1.0    ICMP           172.16.19.1
172.16.19.2
22:27:18  pfe       A      ge-0/0/1.0    ICMP           172.16.19.1
172.16.19.2
22:27:04  pfe       A      lo0.0         ICMP           172.16.17.2
172.16.17.1
22:27:04  pfe       R      ge-0/0/0.0    ICMP           172.16.17.1
172.16.17.2
22:27:04  pfe       A      lo0.0         ICMP           172.16.17.2
172.16.17.1
22:27:04  pfe       R      ge-0/0/0.0    ICMP           172.16.17.1
172.16.17.2
22:27:02  pfe       A      lo0.0         ICMP           172.16.17.2
172.16.17.1
22:27:02  pfe       R      ge-0/0/0.0    ICMP           172.16.17.1
172.16.17.2
22:27:01  pfe       A      lo0.0         ICMP           172.16.17.2
172.16.17.1
22:27:01  pfe       R      ge-0/0/0.0    ICMP           172.16.17.1
172.16.17.2
22:27:00  pfe       A      lo0.0         ICMP           172.16.17.2
172.16.17.1
22:27:00  pfe       R      ge-0/0/0.0    ICMP           172.16.17.1
172.16.17.2
22:24:48  filter_if_group A fxp0.0        ICMP           10.92.16.2
10.92.26.176
```

- To make sure that the firewall filters are active on interface group 1 on Device R1, use the ping *<address>* operational mode command on the CLI of Device R0.

```
user@R0> ping 172.16.17.2
PING 172.16.17.2 (172.16.17.2): 56 data bytes
36 bytes from 172.16.17.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f46b   0 0000  40  01 6239 172.16.17.1  172.16.17.2

36 bytes from 172.16.17.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f479   0 0000  40  01 622b 172.16.17.1  172.16.17.2

36 bytes from 172.16.17.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f487   0 0000  40  01 621d 172.16.17.1  172.16.17.2
```

```
user@R0> ping 20.1.1.2
PING 20.1.1.2 (20.1.1.2): 56 data bytes
36 bytes from 20.1.1.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f5bd   0 0000  40  01 5ae7 20.1.1.1  20.1.1.2

36 bytes from 20.1.1.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f5cd   0 0000  40  01 5ad7 20.1.1.1  20.1.1.2

36 bytes from 20.1.1.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f5d9   0 0000  40  01 5acb 20.1.1.1  20.1.1.2

36 bytes from 20.1.1.2: Communication prohibited by filter
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 f5f6   0 0000  40  01 5aae 20.1.1.1  20.1.1.2
```

- To make sure that the firewall filter is not applied on an interface that is not in interface group 1, use the ping *<address>* operational mode command on the CLI of Device R0.

```
user@R0> ping 172.16.19.2
PING 172.16.19.2 (172.16.19.2): 56 data bytes
```

```
64 bytes from 172.16.19.2: icmp_seq=0 ttl=64 time=8.689 ms
64 bytes from 172.16.19.2: icmp_seq=1 ttl=64 time=4.076 ms
64 bytes from 172.16.19.2: icmp_seq=2 ttl=64 time=8.501 ms
64 bytes from 172.16.19.2: icmp_seq=3 ttl=64 time=3.954 ms
...
```

Meaning

The stateless firewall filter is applied to all interfaces in interface group `1`. The term `term1` match condition in the stateless firewall filter counts, logs, and rejects packets that are received on or sent from the interfaces in interface group `1` and with a source ICMP protocol. The term `term2` match condition matches packets tagged with the ICMP protocol and counts, logs, and accepts those packets. The term `term3` match condition counts all the transit packets.

RELATED DOCUMENTATION

Filtering Packets Received on a Set of Interface Groups Overview | 1373

CHAPTER 21

Configuring Filter-Based Tunneling Across IP Networks

IN THIS CHAPTER

- Understanding Filter-Based Tunneling Across IPv4 Networks | **1394**
- Firewall Filter-Based L2TP Tunneling in IPv4 Networks Overview | **1397**
- Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | **1401**
- Components of Filter-Based Tunneling Across IPv4 Networks | **1403**
- Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | **1409**

Understanding Filter-Based Tunneling Across IPv4 Networks

IN THIS SECTION

- Understanding Filter-Based Tunneling Across IPv4 Networks | **1394**
- Characteristics of Filter-Based Tunneling Across IPv4 Networks | **1395**
- Tunneling with Firewall Filters and Tunneling with Tunnel Interfaces | **1396**

Understanding Filter-Based Tunneling Across IPv4 Networks

Generic routing encapsulation (GRE) in its simplest form is the encapsulation of any network layer protocol over any other network layer protocol to connect disjointed networks that lack a native routing path between them. It is a connectionless and stateless Layer 3 encapsulation protocol, and it offers no mechanisms for reliability, flow control, or sequencing.

GRE tunneling is initiated with standard *firewall filter* actions. Traffic flows through the tunnel provided that the tunnel destination is routable. For MX series routers, this feature is also supported in logical systems.

For MX Series 5G Universal Routing Platforms, when you configure GRE tunneling with firewall filters, you do not need to create tunnel interfaces on Tunnel Services physical interface cards (PICs) or on MPC3E Modular Port Concentrators (MPCs). Instead, PFEs on the Modular Interface Cards (MICs) or MPCs handle the GRE payload encapsulation and decapsulation and provide the tunnel services to the relevant interfaces. As such, a pair of MX Series routers can be installed as provider edge (PE) routers to provide connectivity to customer edge (CE) routers on two disjoint networks.

For PTX Series routers, network services must be set to `enhanced-mode` for filter-based GRE tunneling to work. For more information on filter based tunneling on the PTX, see `tunnel-end-point` .

Ingress Firewall Filter on the Ingress PE Router

On the ingress PE router, you configure a tunnel definition that specifies a unidirectional GRE tunnel. For MX series routers with a MIC or MPC ingress *logical interface*, you attach an encapsulating firewall filter. The firewall filter action references a tunnel definition and initiates the encapsulation of matched packets. The encapsulation process attaches an IPv4 header and a GRE header to the payload packet and then forwards the resulting GRE packet to the filter-specified tunnel.

Ingress Firewall Filter on the Egress PE Router

On the egress PE router, you attach a de-encapsulating firewall filter to the input of all MIC or MPC logical interfaces that are advertised addresses for the router. The firewall filter initiates the de-encapsulation of GRE protocol packets. De-encapsulation removes the inner GRE header and then forwards the original payload packet to its original destination on the destination customer network. If the action specifies an optional routing instance, route lookup is performed using that secondary table instead of the primary table.

Characteristics of Filter-Based Tunneling Across IPv4 Networks

Filter-based tunnels across IPv4 networks are unidirectional. They transport transit packets only, and they do not require tunnel interfaces.

Unidirectional Tunneling

To use filter-based GRE tunnels, start by attaching standard firewall filters at the *input* of each tunnel endpoint (at both the ingress PE router and the egress PE router). At the input to the ingress PE router, you apply an encapsulating firewall filter. At the input to the egress PE router, apply a de-encapsulating firewall filter.

Bidirectional Tunneling

For bidirectional GRE tunneling, you can use the same pair of PE routers, but you must configure a second tunnel in the reverse direction.

Transit Traffic Payloads

A filter-based GRE IPv4 tunnel can transport unicast or multicast transit traffic payloads only. Filter-initiated encapsulation and decapsulation operations execute on PFEs for Ethernet logical interfaces and aggregated Ethernet interfaces. This design enables more efficient use of PFE bandwidth as compared to GRE tunneling using tunnel interfaces. Routing protocol sessions can not be configured on top of the firewall based tunnels.

The PFEs operate in the *forwarding plane* to process packets by forwarding them between input and output interfaces using a locally stored forwarding table, which is a local copy of the information from the Routing Engine (RE).

On the other hand, REs operate in the *control plane* to handle system management, user access to the router, and processes for routing protocols, router interface control, and some chassis component control. The Junos OS architecture separates the functions of these planes to enable flexibility of platform support and scalability of platform performance. Ingress control packets are directed to the control plane where the GRE encapsulation and de-encapsulation processes of the PFEs are not available.

Although you can apply firewall filters to loopback addresses, GRE encapsulating and de-encapsulating firewall filter actions are not supported on router loopback interfaces.

Compact Configuration for Multiple GRE Tunnels

Firewall filters support a wide variety of match criteria and, by extension, the ability to terminate multiple GRE tunnels that match criteria specified in a single firewall filter definition. By creating multiple tunnels, each with its own set of match conditions, you can create tunnels that do not interfere with customer GRE packets or with one another and that re-inject packets to separate routing tables after de-encapsulation.

Tunneling with Firewall Filters and Tunneling with Tunnel Interfaces

Tunneling with tunnel interfaces supports both router control traffic and transit traffic, as well as encryption. Tunneling with firewall filters does not. However, tunneling with firewall filters does provide benefits in performance and scaling.

Forwarding Performance

Filter-based tunneling across IPv4 networks enables more efficient use of PFE bandwidth as compared to GRE tunneling using tunnel interfaces. Encapsulation, de-encapsulation, and route lookup are packet header-processing activities that, for firewall filter-based tunneling, are performed on the PFE. Consequently, the encapsulator never needs to send payload packets to a separate tunnel interface (which might reside on a PIC in a different slot than the interface that receives payload packets).

RELATED DOCUMENTATION

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | **1401**

Components of Filter-Based Tunneling Across IPv4 Networks | **1403**

Firewall Filter Terminating Actions | **884**

tunnel-end-point

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | **1409**

Firewall Filter-Based L2TP Tunneling in IPv4 Networks Overview

IN THIS SECTION

- Unidirectional Tunneling | **1399**
- Tunnel Security | **1399**
- Forwarding Performance | **1400**
- Forwarding Scalability | **1400**

The Layer 2 Tunneling Protocol (L2TP) is a client-server protocol that allows the Point-to-Point Protocol (PPP) to be tunneled across a network. L2TP encapsulates Layer 2 packets, such as PPP, for transmission across a network. An L2TP access concentrator (LAC), configured on an access device, receives packets from a remote client and forwards them to an L2TP network server (LNS) on a remote network. L2TPv3 defines the base control protocol and encapsulation for tunneling multiple Layer 2 connections between two IPv6 nodes. The significant differences between L2TPv2 and L2TPv3 include the following:

- Separation of all PPP-related AVPs and references, which enables the inclusion of a portion of the L2TP data header that was specific to the needs of PPP.

- Transition from a 16-bit Session ID and Tunnel ID to a 32-bit Session ID and Control Connection ID respectively.
- Extension of the tunnel authentication mechanism to cover the entire control message rather than just a portion of certain messages.
- L2TPv3 is supported for IPv6 only.
- For firewall filters, only data plane L2TPv3 encapsulation/ decapsulation is supported.

L2TP is comprised of two types of messages, control messages and data messages (sometimes referred to as control packets and data packets respectively). Control messages are used in the establishment, maintenance, and clearing of control connections and sessions. These messages utilize a reliable control channel within L2TP to guarantee delivery. Data messages are used to encapsulate the L2 traffic being carried over the L2TP session.

You can configure an IPv4 network to transport IPv4, IPv6, or MPLS transit traffic by using GRE tunneling protocol mechanisms initiated by two standard firewall filter actions. This feature is also supported in logical systems. When you configure L2TP tunneling with firewall filters, you do not need to create tunnel interfaces on Tunnel Services physical interface cards (PICs) or on MPC3E Modular Port Concentrators (MPCs). Instead, Packet Forwarding Engines provide tunnel services to Ethernet logical interfaces or aggregated Ethernet interfaces hosted on Modular Interface Cards (MICs) or MPCs in MX Series 5G Universal Routing Platforms.

Two MX Series routers installed as provider edge (PE) routers provide connectivity to customer edge (CE) routers on two disjoint networks. MIC or MPC interfaces on the PE routers perform L2TP IPv4 encapsulation and de-encapsulation of payloads. After decapsulation, packets are sent to the local interface of a routing table specified in the action, or to the default routing table, based on the protocol field of the L2TP header. However, an L2TP packet can optionally be sent across the fabric with a token equal to an output interface index to perform Layer 2 cross- connect. You can specify the output interface specifier to be used for the L2TP packet to be sent by including the `decapsulate l2tp output-interface` *`interface-name`* `cookie` *`l2tpv3-cookie`* statement at the `[edit firewall family` *`family-name`* `filter` *`filter-name`* `term` *`term-name`* `then]` hierarchy level.

During decapsulation, the inner header must be Ethernet for L2TP tunnels. Forwarding class by default is applied before the firewall and it is not preserved for the decapsulated packet (by using the `forwarding-class` *`class-name`* statement at the `[edit firewall family family-name]` hierarchy level, which is a nonterminating filter action). However, you can specify the forwarding class that the packet must be classified against by including the filter action for a decapsulated packet by using the `decapsulate l2tp forwarding-class` *`class-name`* statement at the `[edit firewall family` *`family-name`* `filter filter-name term` *`term-name`* `then]` hierarchy level.

The following field definitions are defined for use in all L2TP Session Header encapsulations.

- The Session ID field is a 32-bit field containing a non-zero identifier for a session. L2TP sessions are named by identifiers that have local significance only. The same logical session will be given different

Session IDs by each end of the control connection for the life of the session. When the L2TP control connection is used for session establishment, Session IDs are selected and exchanged as Local Session ID AVPs during the creation of a session. The Session ID alone provides the necessary context for all further packet processing, including the presence, size, and value of the Cookie, the type of L2-Specific Sublayer, and the type of payload being tunneled.

- The optional Cookie field contains a variable-length value (maximum 64 bits) used to check the association of a received data message with the session identified by the Session ID. The Cookie field must be set to the configured or signaled random value for this session. The Cookie provides an additional level of guarantee that a data message has been directed to the proper session by the Session ID. A well-chosen Cookie might prevent inadvertent misdirection of random packets with recently reused Session IDs or for Session IDs subject to packet corruption. The Cookie might also provide protection against some specific malicious packet insertion attacks. When the L2TP control connection is used for session establishment, random Cookie values are selected and exchanged as Assigned Cookie AVPs during session creation.

A session is a logical connection created between the LAC and the LNS when an end-to-end PPP connection is established between a remote system and the LNS. There is a one-to-one relationship between established L2TP sessions and their associated PPP connections. A tunnel is an aggregation of one or more L2TP sessions.

Starting with Junos OS Release 15.1, decapsulation of IP packets that are sent through an L2TP tunnel with standard firewall filter match conditions and actions specified is performed using a Layer 3 lookup. In Junos OS release 14.2 and earlier, decapsulation of traffic over an L2TP tunnel with firewall filter actions configured is performed using Layer 2 interface properties.

Unidirectional Tunneling

Filter-based L2TP tunnels across IPv4 networks are unidirectional. They transport transit packets only, and they do not require tunnel interfaces. Although you can apply firewall filters to loopback addresses, GRE encapsulating and de-encapsulating firewall filter actions are not supported on router loopback interfaces. Filter-initiated encapsulation and decapsulation operations of L2TP packets execute on Packet Forwarding Engines for Ethernet logical interfaces and aggregated Ethernet interfaces. This design enables more efficient use of Packet Forwarding Engine bandwidth as compared to GRE tunneling using tunnel interfaces. Routing protocol sessions can not be configured on top of the firewall based tunnels.

Tunnel Security

Filter-based tunneling across IPv4 networks is not encrypted. If you require secure tunneling, you must use IP Security (IPsec) encryption, which is not supported on MIC or MPC interfaces. However, Multiservices DPC (MS-DPC) interfaces on MX240, MX480, and MX960 routers support IPsec tools for

configuring manual or dynamic security associations (SAs) for encryption of data traffic as well as traffic destined to or originating at the Routing Engine.

Forwarding Performance

Filter-based tunneling across IPv4 networks enables more efficient use of Packet Forwarding Engine bandwidth as compared to L2TP tunneling using tunnel interfaces. Encapsulation, de-encapsulation, and route lookup are packet header-processing activities that, for firewall filter-based tunneling, are performed on the Junos Trio chipset-based Packet Forwarding Engine. Consequently, the encapsulator never needs to send payload packets to a separate tunnel interface (which might reside on a PIC in a different slot than the interface that receives payload packets).

Forwarding Scalability

Forwarding L2TP traffic with tunnel interfaces requires traffic to be sent to a slot that hosts the tunnel interfaces. When you use tunnel interfaces to forward GRE traffic, this requirement limits the amount of traffic that can be forwarded per GRE tunnel destination address. As an example, suppose you want to send 100 Gbps of L2TP traffic from Router A to Router B and you have only 10 Gbps interfaces. To ensure that your configuration does not encapsulate all the traffic on the same board going to the same 10 Gbps interface, you must distribute the traffic across multiple encapsulation points.

Release History Table

Release	Description
15.1	Starting with Junos OS Release 15.1, decapsulation of IP packets that are sent through an L2TP tunnel with standard firewall filter match conditions and actions specified is performed using a Layer 3 lookup.
14.2	In Junos OS release 14.2 and earlier, decapsulation of traffic over an L2TP tunnel with firewall filter actions configured is performed using Layer 2 interface properties.

RELATED DOCUMENTATION

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | **1401**

Components of Filter-Based Tunneling Across IPv4 Networks | **1403**

Firewall Filter Terminating Actions | **884**

tunnel-end-point

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | **1409**

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks

IN THIS SECTION

- Interfaces on MX240, MX480, MX960, MX2010, and MX2020 Routers | **1401**
- Interfaces on MX5, MX10, MX40, and MX80 Routers | **1401**
- CLI Commit Check for Filter-Based Tunneling Across IPv4 Networks | **1402**

You can attach IPv4 encapsulation and de-encapsulation firewall filters to the input of Ethernet logical interfaces or aggregated Ethernet interfaces hosted on Modular Interface Cards (MICs) or Modular Port Concentrators (MPCs) in MX Series routers.

NOTE: Filter-based generic routing encapsulation (GRE) tunneling is supported on PTX Series routers only when network services is set to `enhanced-mode`. For more information, see `enhanced-mode`.

Interfaces on MX240, MX480, MX960, MX2010, and MX2020 Routers

On MX240, MX480, MX960, MX2010, and MX2020 routers, *firewall filter* actions for IPv4 tunneling are supported on Ethernet logical interfaces or aggregated Ethernet interfaces configured on the following types of ports:

- Ports on MICs that insert into slots in MPCs, which have two Packet Forwarding Engines.
- Ports on a 16-port 10-Gigabit Ethernet MPC (MPC-3D-16XGE-SFPP), a specialized fixed-configuration MPC that has four Packet Forwarding Engines and contains no slots for MICs.

For these physical interfaces, Trio chipset-based Packet Forwarding Engine processes operate in *fabric mode* to provide forwarding and storage functions and lookup and processing functions between Ethernet interfaces and the routing fabric of the chassis.

For information about MPCs, see MX Series MPC Overview and MPCs Supported by MX Series Routers. For information about MICs, see MX Series MIC Overview and MICs Supported by MX Series Routers.

Interfaces on MX5, MX10, MX40, and MX80 Routers

On the MX Series midrange family of routers (MX5, MX10, MX40, and MX80 routers), firewall filter actions for IPv4 tunneling are supported on Ethernet logical interfaces and aggregated Ethernet

interfaces configured on ports on a built-in MIC or on MICs that install into dedicated slots in the router chassis.

- The MX80 router—available as a modular (MX80) or fixed (MX80-48T) chassis—has a built-in 4-port 10-Gigabit Ethernet MIC. The modular chassis has two dedicated slots for MICs. The fixed chassis has 48 built-in tri-rate (10/100/1000Base-T) RJ-45 ports in place of two front-pluggable MIC slots.
- On the MX40 router, only the first two of the four built-in 10-Gigabit Ethernet MIC ports are enabled. As with the modular MX80, the two front-pluggable MIC slots are enabled and support dual-wide MICs that span the two slots.
- The MX5 and MX10 routers are pre-populated with a front-pluggable 20-port Gigabit Ethernet MIC with SFP, and none of the four built-in 10-Gigabit Ethernet MIC ports is enabled. The MX10 supports MICs in both front-pluggable slots, but the MX5 supports MICs in the second slot only.

For more information, see MX5, MX10, MX40, and MX80 Modular Interface Card Description.

The MX Series midrange routers have no switching fabric, and the single Packet Forwarding Engine resides on the base board of the chassis and operates in *standalone mode*. In standalone mode, the Packet Forwarding Engine provides—in addition to forwarding and storage functions and lookup and processing functions—hierarchical queuing, congestion management, and granular statistical functions.

CLI Commit Check for Filter-Based Tunneling Across IPv4 Networks

If you commit a configuration that attaches an encapsulating or de-encapsulating firewall filter to an interface that does not support filter-based tunneling across IPv4 networks, a system event writes a syslog warning message that the interface does not support the filter.

RELATED DOCUMENTATION

Understanding Filter-Based Tunneling Across IPv4 Networks | **1394**

Components of Filter-Based Tunneling Across IPv4 Networks | **1403**

Firewall Filter Terminating Actions | **884**

tunnel-end-point

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | **1409**

Components of Filter-Based Tunneling Across IPv4 Networks

IN THIS SECTION

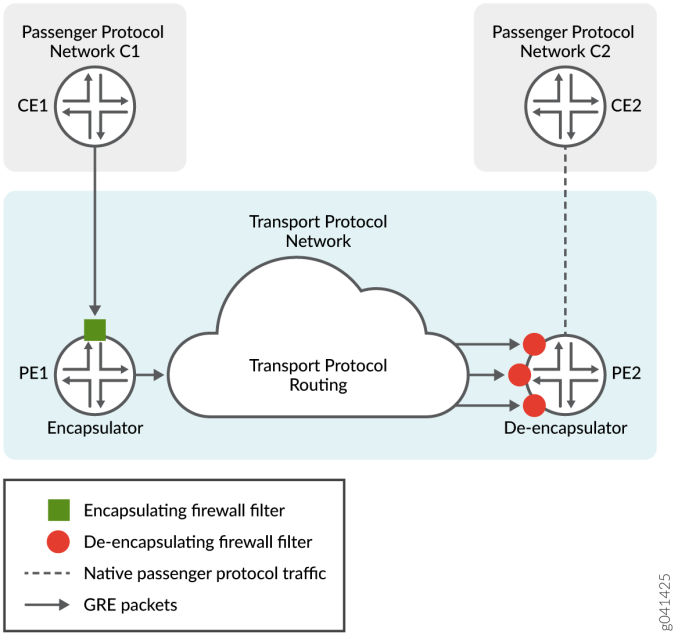
- Topology of Filter-Based Tunneling Across IPv4 Networks | **1403**
- Terminology at the Network Layer Protocols Level | **1405**
- Terminology at the Ingress PE Router | **1405**
- Terminology at the Egress PE Router | **1406**
- GRE Protocol Format for Filter-Based Tunneling Across IPv4 Networks | **1406**

Topology of Filter-Based Tunneling Across IPv4 Networks

NOTE: Filter-based generic routing encapsulation (GRE) tunneling is supported on PTX Series routers only when network services is set to `enhanced-mode`. For more information, see `enhanced-mode`.

Figure 61 on page 1404 shows the path of passenger protocol packets from customer network C1 as they are transported across a service provider IPv4 network to customer network C2.

Figure 61: Unidirectional Filter-Based Tunnel Across an IPv4 Network



In this example topology, C1 and C2 are disjoint networks that lack a native routing path between them. The IPv4 transport network is configured with a unidirectional generic routing encapsulation (GRE) tunnel from PE1 to PE2 using firewall filters and without requiring tunnel interfaces. The GRE tunnel from PE1 to PE2 provides a logical path from C1 to C2 across the IPv4 transport network.

Routing of GRE Packets Across the Tunnel

Traffic flows through the tunnel provided that PE2 is routable from PE1. Routing paths from PE1 to PE2 can be provided by static routes manually added to routing tables or by static or dynamic route-sharing protocols.

Routing of Passenger Protocol Packets from PE2 to C2

By default, PE2 forwards packets based on interface routes (direct routes) imported from the primary routing table. As an option, the de-encapsulating filter can specify that the Packet Forwarding Engine uses an alternate routing table to forward payload packets to the destination customer network. Specify the alternate routing table in a routing instance installed with routes into C2, then use a routing information base (RIB) group definition to share the primary routes with the alternate routes. A RIB group specifies the sharing of routing information (including routes learned from peers, local routes resulting from the application of protocol policies to the learned routes, and the routes advertised to peers) of multiple routing tables.

Terminology at the Network Layer Protocols Level

In filter-based tunneling across an IPv4 network, the network-layer protocols are described in the following terms:

passenger protocol — The type of protocol (IPv4, IPv6, or MPLS) used by the networks that are connected by a GRE tunnel. Packets that are encapsulated and routed across the transport network are *payload packets*.

encapsulation protocol — The type of network layer protocol (GRE) used to encapsulate passenger protocol packets so that the resulting GRE packets can be carried over the transport protocol network as the packet payload.

transport protocol — The type of protocol (IPv4) used by the network that routes passenger protocol packets through a GRE tunnel. The transport protocol is also called the *delivery protocol*.

Terminology at the Ingress PE Router

In filter-based tunneling across an IPv4 network, an egress PE router is described in the following terms:

encapsulator — A PE router that receives packets from a passenger protocol source network, adds an encapsulation protocol (GRE) header and a transport protocol (IPv4) header to this payload, and forwards the resulting GRE packet to the GRE tunnel. This ingress node is also known as the *tunnel source*.

encapsulating interface — On the encapsulator, an Ethernet *logical interface* or an aggregated Ethernet interface configured on a customer-facing interface hosted on a MIC or an MPC. The encapsulating interface receives passenger protocol packets from a CE router. For more information, see "Interfaces That Support Filter-Based Tunneling Across IPv4 Networks" on page 1401.

encapsulation filter — On the encapsulator, a *firewall filter* that you apply to the input of the encapsulating interface. The encapsulating filter action causes the Packet Forwarding Engine to use information in the specified tunnel template to encapsulate matched packets and forward the resulting GRE packets.

tunnel source interface — On the encapsulator, one or more core-facing egress interfaces to the tunnel.

tunnel template — On the encapsulator, a named CLI construct that defines the characteristics of a tunnel:

- Transport protocol family (IPv4).
- IP address or address range of tunnel-facing *egress* interfaces on the encapsulator.
- IP address or address range of tunnel-facing *ingress* interfaces on the de-encapsulator (the egress PE router).
- Encapsulation protocol (GRE).

Terminology at the Egress PE Router

In filter-based tunneling across IPv4 networks, an egress PE router is described in the following terms:

de-encapsulator A PE router that receives GRE packets routed through a filter-based GRE tunnel, removes the transport protocol header and GRE header, and forwards the resulting payload protocol packets to the destination network CE router. The de-encapsulator node is also known as a *de-encapsulating tunnel endpoint* or the *tunnel destination*.

de-encapsulating interfaces On the de-encapsulator, any Ethernet logical interface or aggregated Ethernet interface configured on any core-facing ingress interface that can receive GRE packets from a GRE tunnel. The underlying physical interface must be hosted on a MIC or an MPC. For more information, see "Interfaces That Support Filter-Based Tunneling Across IPv4 Networks" on page 1401.

de-encapsulation filter On the de-encapsulator, a firewall filter that causes the Packet Forwarding Engine to de-encapsulate matched GRE packets and then forward the original passenger protocol packets to destination network CE routers.

GRE packets transported through a single GRE tunnel can arrive at the de-encapsulator node on any of multiple ingress interfaces, depending on how routing is configured. Therefore, you must apply the de-encapsulation firewall filter to the input of every core-facing interface that is an advertised address for the de-encapsulator.

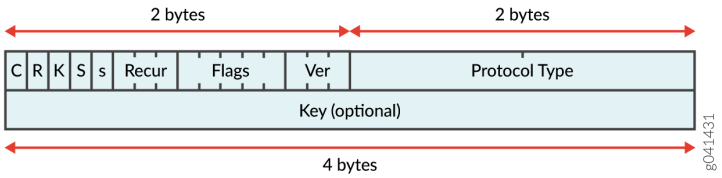
GRE Protocol Format for Filter-Based Tunneling Across IPv4 Networks

In filter-based tunneling across IPv4 networks, the encapsulating interface is an *RFC 1701-compliant transmitter* and the de-encapsulating interfaces are *RFC 1701-compliant receivers*. The packet encapsulation structure implemented in this feature uses a GRE header format that complies with informational RFC 1701, *Generic Routing Encapsulation (GRE)*, October 1994, and with standards track RFC 2784, *Generic Routing Encapsulation (GRE)*, March 2000.

Packet Encapsulation Structure

Filter-based tunneling encapsulates the original passenger protocol packet in an outer shell. For filter-based tunneling across IPv4 networks, the shell adds 24 bytes or 28 bytes of overhead, including 20 bytes of IPv4 header. Figure 62 on page 1407 shows the structure of a passenger protocol packet (the GRE payload) with a GRE header and IPv4 header attached.

Figure 62: Encapsulation Structure for Filter-Based Tunneling Across an IPv4 Network

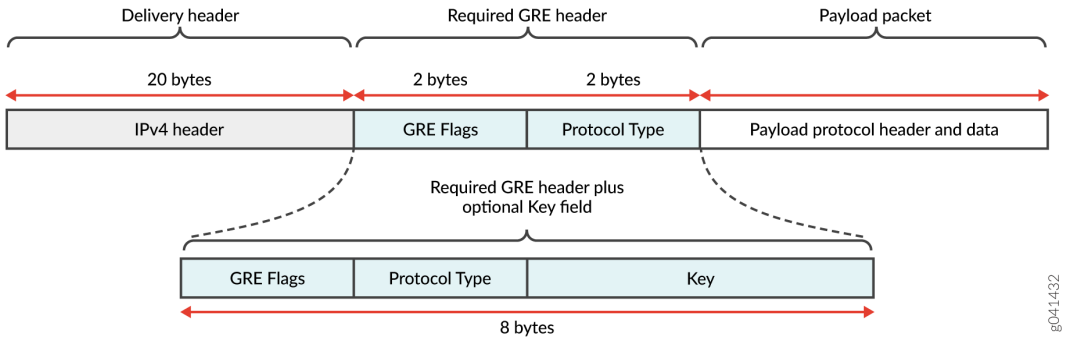


As specified in RFC 1701, five GRE flag bits indicate whether a particular GRE header includes any optional fields (Checksum, Offset, Key, Sequence Number, and Routing). Of the five optional fields, filter-based GRE IPv4 tunneling uses the Key field only.

GRE Header Format

Figure 63 on page 1407 shows the format of the variable-size GRE header used for filter-based tunneling across IPv4 networks, with bit 0 the most significant bit and bit 15 the least significant bit.

Figure 63: GRE Header Format for Filter-Based Tunneling Across IPv4 Networks



The first two octets encode GRE flags, as described in Table 67 on page 1408.

The 2-octet Protocol Type field contains the value 0x0800 to specify the EtherType value for the IPv4 protocol.

The 4-octet Key field is included only if the Key Present bit is set to 1. The Key field carries the key value of the tunnel defined on the encapsulator. If the GRE tunnel definition specifies a key, the Packet Forwarding Engine for the encapsulating endpoint sets the Key Present bit and adds the Key to the GRE header.

Table 67: GRE Flag Values for Filter-Based Tunneling Across IPv4 Networks

Bit Offset and Field Name		Transmitted Value for Filter-Based GRE Tunneling	
0	**C** = Checksum Present	**0**	Checksum field is not used.
1	**R** = Routing Present	**0**	Offset and Routing fields are not used.
2	**K** = Key Present	**0** or **1**	Transmitted as **0** for a keyless tunnel or **1** for a keyed tunnel.
3	**S** = Sequence Number Present	**0**	Sequence Number field is not used.
4	**s** = Strict Source Route	**0**	Not all routing information is Strict Source Routes.
5 - 7	**Recur** = Recursion Control information	**000**	No additional encapsulations are permitted.
8 - 12	**Flags** = Flag bits	**00000**	Reserved.
13 - 15	**Ver** = Version number	**000**	Reserved.

When the Packet Forwarding Engine performs encapsulation for a keyed GRE IPv4 tunnel, the process constructs the first two octets of the GRE header as 0x0000. When the Packet Forwarding Engine performs encapsulation for a non-keyed GRE IPv4 tunnel, the process constructs the first two octets of the GRE header as 0x2000.

RELATED DOCUMENTATION

Understanding Filter-Based Tunneling Across IPv4 Networks | 1394

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | 1401

Firewall Filter Terminating Actions | 884

tunnel-end-point

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling | 1409

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling

IN THIS SECTION

- Requirements | 1409
- Overview | 1411
- Configuration | 1414
- Verification | 1425

This example shows how to configure a unidirectional generic routing encapsulation (GRE) tunnel to transport IPv6 unicast transit traffic across an IPv4 transport network. To provide network connectivity to the two disjoint IPv6 networks, two MX Series 5G Universal Routing Platforms are configured with interfaces that can originate and understand both IPv4 and IPv6 packets. The configuration does not require the creation of tunnel interfaces on Tunnel Services physical interface cards (PICs) or on MPC3E Modular Port Concentrators (MPCs). Instead, you attach firewall filters to Ethernet logical interfaces hosted on Modular Interface Cards (MICs) or MPCs in the two MX Series routers.

NOTE: Filter-based GRE tunneling is supported on PTX Series routers only when network services is set to `enhanced-mode`. For more information, see `enhanced-mode`.

Requirements

This example uses the following Juniper Networks hardware and Junos OS software:

- Transport network—An IPv4 network running Junos OS Release 12.3R2 or later.

- PE routers—Two MX80 routers installed as provider edge (PE) routers that connect the IPv4 network to two disjoint IPv6 networks that require a logical path from one network to the other.
- Encapsulating interface—On the encapsulator (the ingress PE router), one Ethernet logical interface configured on the built-in 10-Gigabit Ethernet MIC.
- De-encapsulating interfaces—On the de-encapsulator (the egress PE router), Ethernet logical interfaces configured on three ports of the built-in 10-Gigabit Ethernet MIC.

Before you begin configuring this example:

1. On each PE router, use the **show chassis fpc pic-status** operational mode command to determine which router line cards support filter-based GRE IPv4 tunneling and then use the `interfaces` configuration statement to configure encapsulating and de-encapsulating interfaces.
   - At PE1, the encapsulator, configure *one encapsulating interface* on a supported line card.
   - At PE2, the de-encapsulator, configure *three de-encapsulating interfaces* on a supported line card.
2. Check that IPv4 routing protocols are enabled across the network to support routing paths from the encapsulator to the de-encapsulator.

   Configure routing information by manually adding static routes to route tables or by configuring static or dynamic route-sharing protocols. For more information, see Transport and Internet Protocols User Guide.
3. At PE1, *ping* the PE2 IPv4 loopback address to verify that the de-encapsulator is reachable from the encapsulator.
4. At PE2, *ping* the CE2 router IPv6 loopback address to verify that the destination customer edge router is reachable from the de-encapsulator..

   IPv6 routing paths from PE2 to CE2 can be provided by static routes manually added to routing tables or by static or dynamic route-sharing protocols.
   - By default, PE2 forwards packets based on interface routes (direct routes) imported from the primary routing table.
   - As an option, the de-encapsulating filter can specify that the Packet Forwarding Engine uses an alternate routing table to forward payload packets to the destination customer network. In an optional configuration task in this example, you specify an alternate routing table by installing static routes from PE2 to C1 in the routing instance **blue**. You configure the routing information base (RIB) group **blue_group** to specify that the route information of **inet6.0** is shared with **blue.inet6.0**, then you associate the PE2 interfaces with routes stored in both the default routes and the routing instance.

Overview

IN THIS SECTION

- Topology | 1411

In this example you configure a unidirectional filter-based GRE IPv4 tunnel from Router PE1 to Router PE2, providing a logical path from IPv6 network C1 to IPv6 network C2.

NOTE: To enable *bidirectional* filter-based GRE tunneling, you must configure a second tunnel in the reverse direction.

As an optional task in this example, you can create a RIB group, which specifies the sharing of routing information (including routes learned from peers, local routes resulting from the application of protocol policies to the learned routes, and the routes advertised to peers) of multiple routing tables.

Topology

Figure 64 on page 1412 shows the path of IPv6 traffic transported from network C1 to network C2, across an IPv4 transport network using a filter-based tunnel from PE1 to PE2 and without requiring tunnel interfaces.

Figure 64: Filter-Based Tunnel from PE1 to PE2 in an IPv4 Network

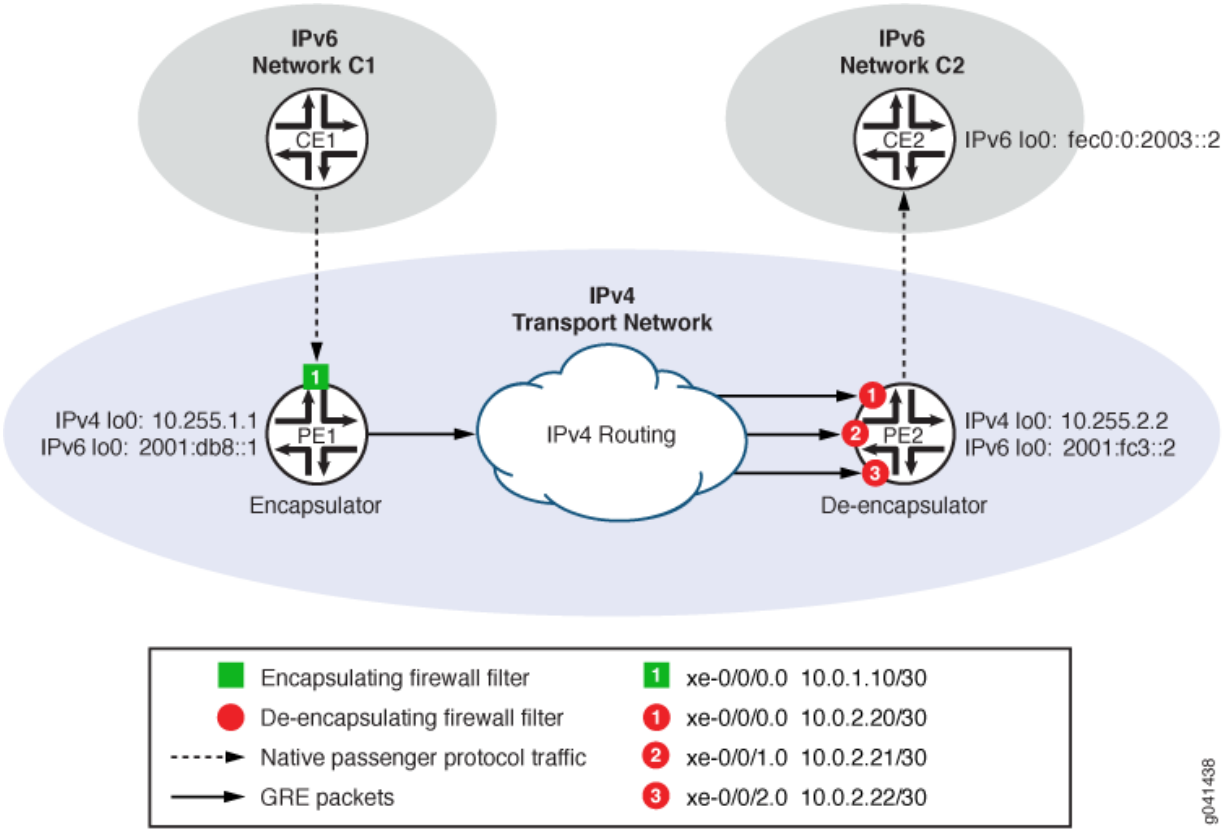


Table 68 on page 1412 summarizes the configuration of Router PE1 as the encapsulator. Table 69 on page 1413 summarizes the configuration of Router PE2 as the de-encapsulator.

Table 68: Encapsulator Components on PE1

Component	CLI Names		Description
Encapsulator	Device name: IPv4 loopback: IPv6 loopback:	PE1 10.255.1.1 2001:db8::1	MX80 router installed as an ingress PE router. PE1 connects the IPv4 network the customer edge router CE1 in the IPv6 source network C1.
Encapsulating interface	Interface name: IPv4 address: IPv6 address:	xe-0/0/0.0 10.0.1.10/30 ::10.34.1.10/120	Customer-facing logical interface hosted on a 10-Gigabit Ethernet MIC. CE1 sends this interface IPv6 traffic that originates at end-user hosts and is destined for applications or hosts on the IPv6 destination network C2.

Table 68: Encapsulator Components on PE1 *(Continued)*

Component	CLI Names		Description
Encapsulation filter	Filter name:	gre_encap_1	IPv6 firewall filter whose action causes the Packet Forwarding Engine to encapsulate matched packets using the specified tunnel characteristics. Encapsulation consists of adding a GRE header, adding an IPv4 packet header, and then forwarding the resulting GRE packet through the GRE IPv4 tunnel.
Tunnel source interface	Interface name: IPv4 address:	xe-0/0/2.0 10.0.1.12	Core-facing egress interface to the tunnel.
GRE tunnel template	Tunnel name:	tunnel_1	Defines the GRE IPv4 tunnel from Router PE1 (10.255.1.1) to Router PE2(10.255.2.2), using the tunneling protocol supported on IPv4 (gre).

Table 69: De-Encapsulator Components on PE2

Component	CLI Names		Description
De-encapsulator	Device name: IPv4 loopback: IPv6 loopback:	PE2 10.255.2.2 2001:fc3::2	MX80 router installed as an egress PE router to receive GRE packets forwarded from ingress router PE1 across a GRE IPv4 tunnel.
De-encapsulating interfaces	Interface name: IPv4 address: Interface name: IPv4 address: Interface name: IPv4 address:	xe-0/0/0.0 10.0.2.24/30 xe-0/0/1.0 10.0.2.21/30 xe-0/0/2.0 10.0.2.22/30	Core-facing ingress logical interfaces hosted on 10-Gigabit Ethernet MICs. The interfaces receive GRE packets routed through the GRE IPv4 tunnel from PE1.

Table 69: De-Encapsulator Components on PE2 *(Continued)*

Component	CLI Names		Description
De-encapsulation filter	Filter name:	`gre_decap_1`	IPv4 firewall filter that applies the **decapsulate** action to GRE packets. The filter action causes the Packet Forwarding Engine to de-encapsulate matched packets. De-encapsulation consists of removing the outer GRE header and then forwarding the inner IPv6 payload packet to its original destination on the destination IPv6 network by performing destination lookup on the default routing table.

Tunnel egress interface	Interface name:
	IPv4 address:
	IPv6 address:
		`xe-0/0/3.0`
		`10.0.2.23/30`
		`::20.34.2.23/120`
			Customer-facing interface though which the router forwards de-encapsulated IPv6 packets to the destination IPv6 network C2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1414**
- Configuring PE1 to Encapsulate IPv6 Packets | **1416**
- Configuring PE2 to De-Encapsulate GRE Packets | **1419**
- Optional: Configuring PE2 with an Alternate Routing Table | **1423**

To transport IPv6 packets from CE1 to CE2 across an IPv4 transport network using a filter-based tunnel from PE1 to PE2 and without configuring tunnel interfaces, perform these tasks:

CLI Quick Configuration

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

Configuring PE1 to Encapsulate IPv6 Packets

```
set interfaces lo0 unit 0 family inet address 10.255.1.1
set interfaces lo0 unit 0 family inet6 address 2001:db8::1
set interfaces xe-0/0/0 unit 0 family inet address 10.0.1.10/30
set interfaces xe-0/0/0 unit 0 family inet6 address 2001::10.34.1.10/120
set interfaces xe-0/0/0 unit 0 family inet6 filter input gre_encap_1
set interfaces xe-0/0/2 unit 0 family inet address 10.0.1.12/30
set firewall family inet6 filter gre_encap_1 term t1 then count c_gre_encap_1
set firewall family inet6 filter gre_encap_1 term t1 then encapsulate tunnel_1
set firewall tunnel-end-point tunnel_1 ipv4 source-address 10.255.1.1
set firewall tunnel-end-point tunnel_1 ipv4 destination-address 10.255.2.2
set firewall tunnel-end-point tunnel_1 gre
```

Configuring PE2 to De-Encapsulate GRE Packets

```
set interfaces lo0 unit 0 family inet address 10.255.2.2
set interfaces lo0 unit 0 family inet6 address 2001:fc3::2
set interfaces xe-0/0/0 unit 0 family inet address 10.0.2.20/30
set interfaces xe-0/0/1 unit 0 family inet address 10.0.2.21/30
set interfaces xe-0/0/2 unit 0 family inet address 10.0.2.22/30
set interfaces xe-0/0/3 unit 0 family inet address 10.0.2.23/30
set interfaces xe-0/0/3 unit 0 family inet6 address ::20.34.2.23/120
set forwarding-options family inet filter input gre_decap_1
set firewall family inet filter gre_decap_1 term t1 from source-address 10.255.1.1/32
set firewall family inet filter gre_decap_1 term t1 from destination-address 10.255.2.2/32
set firewall family inet filter gre_decap_1 term t1 then count c_gre_decap_1
set firewall family inet filter gre_decap_1 term t1 then decapsulate gre
```

Optional: Configuring PE2 with an Alternate Routing Table

```
set routing-instances blue instance-type forwarding
set routing-instances blue routing-options rib blue.inet6.0 static route 0::/0 next-hop
fec0:0:2003::2
set routing-options passive
set routing-options rib inet6.0
set routing-options rib-groups blue_group import-rib inet6.0
set routing-options rib-groups blue_group import-rib blue.inet6.0
```

```
set routing-options interface-routes rib-group inet6 blue_group
set firewall family inet filter gre_decap_1 term t1 then decapsulate gre routing-instance blue
```

Configuring PE1 to Encapsulate IPv6 Packets

Step-by-Step Procedure

To configure Router PE1 to encapsulate IPv6 packets arriving from CE1:

1. Configure the router loopback addresses.

```
[edit]
user@PE1# set interfaces lo0 unit 0 family inet address 10.255.1.1
user@PE1# set interfaces lo0 unit 0 family inet6 address 2001:db8::1
```

2. Configure the encapsulating interface IPv4 and IPv6 addresses and attach the encapsulating filter to the IPv6 input.

```
[edit]
user@PE1# set interfaces xe-0/0/0 unit 0 family inet address 10.0.1.10/30
user@PE1# set interfaces xe-0/0/0 unit 0 family inet6 address ::10.34.1.10/120
user@PE1# set interfaces xe-0/0/0 unit 0 family inet6 filter input gre_encap_1
```

3. Configure the core-facing egress interface to the tunnel.

```
[edit]
user@PE2# set interfaces xe-0/0/2 unit 0 family inet address 10.0.1.12/30
```

4. Define an IPv6 firewall filter that causes the Packet Forwarding Engine to encapsulate all packets.

```
[edit]
user@PE1# set firewall family inet6 filter gre_encap_1 term t1 then count c_gre_encap_1
user@PE1# set firewall family inet6 filter gre_encap_1 term t1 then encapsulate tunnel_1
```

NOTE: The **encapsulate** firewall filter action is a *terminating* filter action. A filter-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.

5. Define a GRE IPv4 tunnel template named tunnel_1 that specifies the host IP addresses of the one tunnel source interface and three tunnel destination interfaces.

```
[edit]
user@PE1# set firewall tunnel-end-point tunnel_1 ipv4 source-address 10.255.1.1
user@PE1# set firewall tunnel-end-point tunnel_1 ipv4 destination-address 10.255.2.2
user@PE1# set firewall tunnel-end-point tunnel_1 gre
```

NOTE: You can tunnel multiple but distinct flows from 10.0.1.10 (the tunnel source interface on PE1) to 10.0.2.20 – 10.0.2.22 (the de-encapsulating interfaces on PE2) if you use the GRE option **key *number*** to uniquely identify each tunnel.

6. If you are done configuring the device, commit the configuration.

```
[edit ]
user@PE1# commit
```

Results

From configuration mode, confirm your configuration by entering the `show firewall` and `show interfaces` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE1

Confirm the firewall filter and tunnel template on the encapsulator.

```
user@PE2# show firewall
family inet6 {
    filter gre_encap_1 {
        term t1 {
            then {
                count c_gre_encap_1;
```

```
                encapsulate tunnel_1;
            }
        }
    }
}
tunnel-end-point tunnel_1 {
    ipv4 {
        source-address 10.255.1.1;
        destination-address 10.255.2.2;
    }
    gre;
}
```

Router PE1

Confirm the interfaces on the encapsulator.

```
user@PE1# show interfaces
lo0 {
    unit 0 {
        family inet {
            address 10.255.1.1;
        }
        family inet6 {
            address 2001:db8::1;
        }
    }
}
xe-0/0/0 {
    unit 0 {
        family inet {
            address 10.0.1.10/30;
        }
        family inet6 {
            address ::10.34.1.10/120;
            filter input gre_encap_1;
        }
    }
}
xe-0/0/2 {
    unit 0 {
        family inet {
```

```
            address 10.0.1.12/30;
        }
    }
}
```

Configuring PE2 to De-Encapsulate GRE Packets

Step-by-Step Procedure

To configure Router PE2 to de-encapsulate GRE packets arriving from the IPv4 tunnel:

1. Configure the router loopback address.

```
[edit]
user@PE2# set interfaces lo0 unit 0 family inet address 10.255.2.2
user@PE2# set interfaces lo0 unit 0 family inet6 address 2001:fc3::2
```

2. Configure the de-encapsulating interfaces.

```
[edit]
user@PE2# set interfaces xe-0/0/0 unit 0 family inet address 10.0.2.20/30
user@PE2# set interfaces xe-0/0/1 unit 0 family inet address 10.0.2.21/30
user@PE2# set interfaces xe-0/0/2 unit 0 family inet address 10.0.2.22/30
```

3. Configure the customer-facing egress interface to CE2.

```
[edit]
user@PE2# set interfaces xe-0/0/3 unit 0 family inet address 10.0.2.23/30
user@PE2# set interfaces xe-0/0/3 unit 0 family inet6 address ::20.34.2.23/120
```

4. Apply the ingress de-encapsulating firewall filter to all forwarded packets.

```
[edit]
user@PE2# set forwarding-options family inet filter input gre_decap_1
```

5. Define IPv4 filter **gre_decap_1**.

Define an IPv4 filter that de-encapsulates and forwards all GRE packets.

```
[edit]
user@PE2# set firewall family inet filter gre_decap_1
```

6. Configure term **t1** to match packets transported across the tunnel **tunnel_1** defined on Router PE1. The tunnel sends packets from Router PE1 (configured with IPv4 loopback address 10.255.1.1) to Router PE2 (configured with IPv4 loopback address 10.255.2.2).

```
[edit firewall family inet filter gre_decap_1]
user@PE2# set term t1 from source-address 10.255.1.1
user@PE2# set  term t1 from destination-address 10.255.2.2
```

7. Configure term **t1** to count and de-encapsulate matched packets.

```
[edit firewall family inet filter gre_decap_1]
user@PE2# set term t1 then count c_gre_decap_1
user@PE2# set term t1 then decapsulate gre
```

If the de-encapsulating filter action **decapsulate** references the **blue** routing instance, make sure that the routing instance is configured and that the RIB group **blue_group** defines the sharing of the alternate routes into the primary table.

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE2# commit
```

Results

From configuration mode, confirm your configuration by entering the `show firewall`, `show forwarding-options`, and `show interfaces` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE2

Confirm the firewall filter on the de-encapsulator.

```
user@PE2# show firewall
family inet {
    filter gre_decap_1 {
        term t1 {
            from {
                source-address 10.255.1.1;
                destination-address 10.255.2.2;
            }
            then {
                count c_gre_decap_1;
                decapsulate gre routing-instance blue;
            }
        }
    }
}
```

NOTE: If the de-encapsulating filter action **decapsulate** references the **blue** routing instance, make sure that the routing instance is configured and that the RIB group **blue_group** defines the sharing of the alternate routes into the primary table.

Router PE2

Confirm the forwarding options (for attaching the de-encapsulating firewall filter to all input forwarded packets) on the de-encapsulator.

```
user@PE2# show forwarding-options
forwarding-options {
    family inet {
        filter {
            input gre_decap_1;
        }
    }
}
```

Router PE2

Confirm the interfaces on the de-encapsulator.

```
user@PE2# show interfaces
lo0 {
    unit 0 {
        family inet {
            address 10.255.2.2;
        }
        family inet6 {
            address 2001:fc3::2;
        }
    }
}
xe-0/0/0 {
    unit 0 {
        family inet {
            address 10.0.2.20/30;
            filter input gre_decap_1;
        }
    }
}
xe-0/0/1 {
    unit 0 {
        family inet {
            address 10.0.2.21/30;
            filter input gre_decap_1;
        }
    }
}
xe-0/0/2 {
    unit 0 {
        family inet {
            address 10.0.2.22/30;
            filter input gre_decap_1;
        }
    }
}
xe-0/0/3 {
    unit 0 {
        family inet {
            address 10.0.2.23/30;
        }
```

```
        family inet6 {
            address ::20.34.2.23/120;
        }
    }
}
```

Optional: Configuring PE2 with an Alternate Routing Table

Step-by-Step Procedure

To configure Router PE2 with an alternate routing table:

1. Configure the routing instance **blue**, and add static routes to CE2.

```
[edit ]
user@PE2# set routing-instances blue instance-type forwarding
user@PE2# set routing-instances blue routing-options rib blue.inet6.0 static route 0::/0 next-hop fec0:0:2003::2
```

   The Junos OS software generates the routing table **blue.inet6.0** using the routing information learned within the instance.

2. Enable routes to remain in routing and forwarding tables, even if the routes become inactive. This allows a static route to remain in the table if the next hop is unavailable.

```
[edit ]
user@PE2# set routing-options passive
```

3. Create a RIB group by explicitly creating the default routing table.

```
[edit ]
user@PE2# set routing-options rib inet6.0
```

4. Define the RIB group **blue_group**.

```
[edit ]
user@PE2# set routing-options rib-groups blue_group import-rib inet6.0
user@PE2# set routing-options rib-groups blue_group import-rib blue.inet6.0
```

In the **import-rib** statement, specify the primary routing table first.

5. Associate the router interfaces with routing information specified by the RIB group.

```
[edit ]
user@PE2# set routing-options interface-routes rib-group inet6 blue_group
```

6. If you are done configuring the device, commit the configuration.

```
[edit ]
user@PE2# commit
```

Results

From configuration mode, confirm your configuration by entering the `show firewall`, `show routing-instances`, and `show routing-options`commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE2

If you configured an alternate routing table on Router PE2, confirm the routing instance configuration.

```
user@PE2# show routing-instances
blue {
    instance-type forwarding;
    routing-options {
        static route 0::/0 next-hop fec0:0:2003::2;
    }
}
```

Router PE2

If you configured an alternate routing table on Router PE2, confirm the RIB group and direct routing configurations.

```
user@PE2# show routing-options
interface-routes {
    rib-group blue_group;
}
passive;
```

```
rib inet6.0;
rib-groups {
    blue_group {
        import-rib [ inet6.0 blue.inet6.0 ];
    }
}
```

Verification

IN THIS SECTION

- Verifying Routing Information | **1425**
- Verifying Encapsulation on PE1 | **1427**
- Verifying De-Encapsulation on PE2 | **1428**

Confirm that the configurations are working properly.

Verifying Routing Information

Purpose

Verify that the direct routes include the alternate routing table information.

Action

To perform the verification:

1. (Optional) To verify the routing instance **blue** on PE2, use the **show route instance** operational mode command to display the primary table and number of routes for that routing instance.

```
user@PE2> show route instance blue summary
Instance             Type
         Primary RIB                                    Active/holddown/hidden
blue                 forwarding
         blue.inet6.0                                   2/0/0
```

2. (Optional) To view the routing table associated with the routing instance **blue** on PE2, use the **show route table** operational mode command

```
user@PE2> show route table blue.inet6.0

blue.inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::192:168:239:17/128
                   *[Direct/0] 00:02:26
                    > via lo0.0
fe80::2a0:a50f:fc64:e032/128
                   *[Direct/0] 00:02:26
                    > via lo0.0
```

3. (Optional) To verify that the alternate routes from routing instance blue have been imported to the PE2 forwarding table, use the **show route forwarding-table** operational mode command to display the contents of the router forwarding table and the routing instance forwarding table.

```
user@PE2> show route forwarding-table blue
Routing table: blue.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   689     1
0.0.0.0/32         perm     0                    dscd   687     1
172.16.233.0/4        perm     0                    mdsc   688     1
172.16.233.1/32       perm     0 172.16.233.1          mcst   684     1
255.255.255.255/32 perm     0                    bcst   685     1

Routing table: blue.iso
ISO:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   695     1

Routing table: blue.inet6
Internet6:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   701     1
::/128             perm     0                    dscd   699     1
2001:db8::192:168:239:17/128
                   user     0                    rtbl     2     3
```

```
fe80::2a0:a50f:fc64:e032/128
                   user     0                    rtbl     2     3
ff00::/8           perm     0                    mdsc   700     1
ff02::1/128        perm     0 ff02::1            mcst   697     1
```

Verifying Encapsulation on PE1

Purpose

Verify the encapsulating interface on PE1.

Action

To perform the verification:

1. Use the **show interfaces filters** operational mode command to verify that the encapsulating firewall filter is attached to the ingress of the encapsulating interface.

```
user@PE1> show interfaces filters xe-0/0/0.0
Interface        Admin  Link  Proto  Input Filter        Output Filter
xe-0/0/0.0       up     down  inet6  gre_encap_1
```

2. Use the **show interfaces** operational mode command to verify that the encapsulating interface is receiving packets.

```
user@PE1> show interfaces xe-0/0/0.0 detail | filter ”Ingress traffic”
...
Physical interface: xe-0/0/0, Enabled, Physical link is Up
...
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :           6970299398                   0 bps
   Input  packets:             81049992                   0 pps
   Drop   bytes  :                    0                   0 bps
   Drop   packets:                    0                   0 pps
...
```

3. Use the **show firewall filter** operational mode command to verify that ingress passenger protocol traffic triggers the encapsulating filter.

```
user@PE1> show firewall filter gre_encap_1
Filter: gre_encap_1
Counters:
Name                              Bytes             Packets
c_gre_encap_1               6970299398            81049992
```

Meaning

If the encapsulating filter is attached to the encapsulating interface, and the encapsulating interface receives passenger protocol traffic, and the firewall filter statistics show that ingress passenger protocol traffic is being encapsulated, then GRE packets are being forwarded through the tunnel.

Verifying De-Encapsulation on PE2

Purpose

Verify the de-encapsulating interfaces on PE2.

Action

To perform the verification:

1. On PE1, use the **ping** operational mode command to verify that PE2 is reachable.

```
user@PE1> ping 10.255.2.2
PING 10.255.2.2 (10.255.2.2): 56 data bytes
64 bytes from 10.255.2.2: icmp_seq=0 ttl=64 time=0.576 ms
64 bytes from 10.255.2.2: icmp_seq=1 ttl=64 time=0.269 ms
^C [abort]
```

2. On PE2, use the **show interfaces filter** operational mode command to verify that the de-encapsulating firewall filter is attached to the ingress of the de-encapsulating interfaces.

```
user@PE2> show interfaces filter | match xe-
Interface        Admin  Link  Proto  Input Filter        Output Filter
xe-0/0/0.0       up     down  inet   gre_decap_1
```

```
xe-0/0/1.0       up     down  inet   gre_decap_1
xe-0/0/2.0       up     down  inet   gre_decap_1
```

3. On PE2, use the **show interfaces** operational mode command to verify that the de-encapsulating interfaces are receiving packets.

```
user@PE2> show interfaces xe-0/0/0.0 detail | filter ”Ingress traffic”
Physical interface: xe-0/0/0, Enabled, Physical link is Up
 ...
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :           6970299398                   0 bps
   Input  packets:             81049992                   0 pps
   Drop   bytes  :                    0                   0 bps
   Drop   packets:                    0                   0 pps
 ...


user@PE2> show interfaces xe-0/0/1.0 detail | filter ”Ingress traffic”
Physical interface: xe-0/0/2, Enabled, Physical link is Up
 ...


user@PE2> show interfaces xe-0/0/2.0 detail | filter ”Ingress traffic”
Physical interface: xe-0/0/2, Enabled, Physical link is Up
 ...
```

Depending on how routing is configured and which links are up and which links are down, some of the de-encapsulating interfaces might not be receiving packets although the tunnel is operating properly.

4. On PE2, use the **show firewall filter** operational mode command to verify that ingress GRE traffic triggers the de-encapsulating filter.

```
user@PE2> show firewall filter gre_decap_1

Filter: gre_decap_1
Counters:
Name                                                Bytes              Packets
c_gre_decap_1                                  6970299398             81049992
```

Meaning

The verification confirms the following operational states and activities of the encapsulator:

- PE2 is reachable from the PE1.
- The de-encapsulating filter is attached to the input of all de-encapsulating interfaces.
- The de-encapsulator is receiving traffic at de-encapsulating interfaces as expected.
- GRE packets received at the de-encapsulating interfaces trigger the de-encapsulating firewall filter action.

RELATED DOCUMENTATION

Understanding Filter-Based Tunneling Across IPv4 Networks | **1394**

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks | **1401**

Components of Filter-Based Tunneling Across IPv4 Networks | **1403**

Firewall Filter Terminating Actions | **884**

tunnel-end-point

clear firewall

show chassis fpc

show firewall

show firewall log

show interfaces (Aggregated Ethernet)

show interfaces

show route forwarding-table

Junos OS Support for IPv4, IPv6, and MPLS Routing Protocols

CHAPTER 22

Configuring Service Filters

IN THIS CHAPTER

- Service Filter Overview | 1431
- How Service Filters Evaluate Packets | 1433
- Guidelines for Configuring Service Filters | 1435
- Guidelines for Applying Service Filters | 1438
- Example: Configuring and Applying Service Filters | 1441
- Service Filter Match Conditions for IPv4 or IPv6 Traffic | 1449
- Service Filter Nonterminating Actions | 1460
- Service Filter Terminating Actions | 1461

Service Filter Overview

IN THIS SECTION

- Services | 1431
- Service Rules | 1432
- Service Rule Refinement | 1432
- Service Filter Counters | 1432

Services

The Adaptive Services Physical Interface Cards (PICs), Multiservices PICs, and Multiservices Dense Port Concentrators (DPCs) provide *adaptive services interfaces*. Adaptive services interfaces enable you to coordinate a special range of services on a single PIC or DPC by configuring a set of services and applications.

NOTE: Service filters are not supported on T4000 routers.

Service Rules

A *service set* is an optional definition you can apply to the traffic at an adaptive services interface. A service set enables you to configure combinations of directional rules and default settings that control the behavior of each service in the service set.

Service Rule Refinement

When you apply a service set to the traffic at an adaptive services interface, you can optionally use *service filters* to refine the target of the set of services and also to process traffic. Service filters enable you to manipulate traffic by performing packet filtering to a defined set of services on an adaptive services interface before the traffic is delivered to its destination. You can apply a service filter to traffic before packets are accepted for input or output service processing or after packets return from input service processing.

Service Filter Counters

Like standard firewall filters, service filters support counting of matched packets. When you display counters for a service filter, however, the syntax for specifying the filter name includes the name of the *service set* to which the service filter is applied.

- To enable counting of the packets matched by a service filter term, specify the `count` *`counter-name`* nonterminating action in that term.
- To display counters for service filters, use the `show firewall filter` *`filter-name`* `<counter` *`counter-name`*`>` *operational mode command*, and specify the *`filter-name`* as follows:

  ```
  __service-service-set-name:service-filter-name
  ```

For example, suppose you configure a service filter named `out_filter` with a counter named `out_counter` and apply that service filter to a *logical interface* to direct certain packets for processing by the output

services associated with the service set `nat_set`. In this scenario, the syntax for using the `show firewall` operational mode command to display the counter is as follows:

```
[edit]
user@host> show firewall filter __service-nat_set:out_filter counter out_counter
```

RELATED DOCUMENTATION

- Stateless Firewall Filter Types
- How Service Filters Evaluate Packets | 1433
- Guidelines for Configuring Service Filters | 1435
- Guidelines for Applying Service Filters | 1438
- Example: Configuring and Applying Service Filters | 1441
- *Adaptive Services and Multiservices Interfaces Overview*
- *Configuring Service Sets to be Applied to Services Interfaces*
- *Configuring Service Rules*

How Service Filters Evaluate Packets

IN THIS SECTION

- Service Filters That Contain a Single Term | 1433
- Service Filters That Contain Multiple Terms | 1434
- Service Filter Terms That Do Not Contain Any Match Conditions | 1434
- Service Filter Terms That Do Not Contain Any Actions | 1434
- Service Filter Default Action | 1434

Service Filters That Contain a Single Term

For a service filter that consists of a single term, the policy framework software evaluates a packet as follows:

- If the packet matches all the conditions, the actions are taken.
- If the packet matches all the conditions and no actions are specified, the packet is accepted.
- If the packet does not match all the conditions, it is discarded.

Service Filters That Contain Multiple Terms

For a service filter that consists of multiple terms, the policy framework software evaluates a packet against the terms in the filter sequentially, beginning with the first term in the filter, until either the packet matches all the conditions in one of the terms or there are no more terms in the filter.

- If the packet matches all the conditions in a term, the actions in that term are performed and evaluation of the packet ends at that term. Any subsequent terms in the filter are not used.
- If the packet does not match all the conditions in the term, evaluation of the packet proceeds to the next term in the filter.

Service Filter Terms That Do Not Contain Any Match Conditions

For service filters with a single term and for filters with multiple terms, if a term does not contain any match conditions, the actions are taken on any packet evaluated.

Service Filter Terms That Do Not Contain Any Actions

If a term does not contain any actions, and if the packet matches the conditions in the term, the packet is accepted.

Service Filter Default Action

Each service filter has an *implicit* `skip` action at the end of the filter, which is equivalent to including the following example term `explicit_skip` as the final term in the service filter:

```
term explicit_skip {
    then skip;
}
```

By default, if a packet matches none of the terms in a service filter, the packet bypasses service processing.

RELATED DOCUMENTATION

Service Filter Overview | 1431
Guidelines for Configuring Service Filters | 1435
Guidelines for Applying Service Filters | 1438
Example: Configuring and Applying Service Filters | 1441

Guidelines for Configuring Service Filters

IN THIS SECTION

- Statement Hierarchy for Configuring Service Filters | 1435
- Service Filter Protocol Families | 1436
- Service Filter Names | 1436
- Service Filter Terms | 1436
- Service Filter Match Conditions | 1436
- Service Filter Terminating Actions | 1437

Statement Hierarchy for Configuring Service Filters

To configure a service filter, include the `service-filter` *`service-filter-name`* statement at the `[edit firewall family (inet | inet6)]` hierarchy level:

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

```
        }
    }
}
```

Individual statements supported under the `service-filter` *`service-filter-name`* statement are described separately in this topic and are illustrated in the example of configuring and applying a service filter.

Service Filter Protocol Families

You can configure service filters to filter IPv4 traffic (`family inet`) and IPv6 traffic (`family inet6`) only. No other protocol families are supported for service filters.

Service Filter Names

Under the `family inet` or `family inet6` statement, you can include `service-filter` *`service-filter-name`* statements to create and name service filters. 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 (“ ”).

Service Filter Terms

Under the `service-filter` *`service-filter-name`* statement, you can include `term` *`term-name`* statements to create and name filter terms.

- You must configure at least one term in a *firewall filter*.
- You must specify a unique name for each term within a firewall filter. The term name 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 (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter 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` configuration mode command to reorder the terms of a firewall filter.

Service Filter Match Conditions

Service filter terms support only a subset of the IPv4 and IPv6 match conditions that are supported for standard stateless firewall filters.

If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing*

Architecture. For more information about IPv6 addresses, see "IPv6 Overview" in the Junos OS Routing Protocols Library for Routing Devices.

Service Filter Terminating Actions

When configuring a service filter term, you must specify one of the following filter-terminating actions:

- `service`
- `skip`

> **NOTE**: These actions are unique to service filters.

Service filter terms support only a subset of the IPv4 and IPv6 nonterminating actions that are supported for standard stateless firewall filters:

- `count` *`counter-name`*
- `log`
- `port-mirror`
- `sample`

Service filters do not support the **next** action.

RELATED DOCUMENTATION

Service Filter Overview | **1431**

How Service Filters Evaluate Packets | **1433**

Guidelines for Applying Service Filters | **1438**

Service Filter Match Conditions for IPv4 or IPv6 Traffic | **1449**

Service Filter Terminating Actions | **1461**

Service Filter Nonterminating Actions | **1460**

Example: Configuring and Applying Service Filters | **1441**

Guidelines for Applying Service Filters

IN THIS SECTION

- Restrictions for Adaptive Services Interfaces | **1438**
- Statement Hierarchy for Applying Service Filters | **1439**
- Associating Service Rules with Adaptive Services Interfaces | **1439**
- Filtering Traffic Before Accepting Packets for Service Processing | **1440**
- Postservice Filtering of Returning Service Traffic | **1441**

Restrictions for Adaptive Services Interfaces

The following restrictions apply to adaptive services interfaces and service filters.

Adaptive Services Interfaces

You can apply a service filter to IPv4 or IPv6 traffic associated with a service set at an *adaptive services interface* only. Adaptive services interfaces are supported for the following hardware only:

- Adaptive Services (AS) PICs on M Series and T Series routers
- Multiservices (MS) PICs on M Series and T Series routers
- MS DPCs on MX Series routers and EX Series switches
- MS MPCs and MICs on MX Series routers

System Logging to a Remote Host from M Series Routers

Logging of adaptive services interfaces messages to an external server by means of the `fxp0` or `em0` port is not supported on M Series routers. The architecture does not support system logging traffic out of a management interface. Instead, access to an external server is supported on a Packet Forwarding Engine interface.

Statement Hierarchy for Applying Service Filters

You can enable packet filtering of IPv4 or IPv6 traffic before a packet is accepted for input or output service processing. To do this, apply a service filter to the adaptive services interface input or output in conjunction with an interface service set.

You can also enable packet filtering of IPv4 or IPv6 traffic that is returning to the Packet Forwarding Engine after input service processing completes. To do this, apply a post-service filter to the adaptive services interface input.

The following configuration shows the hierarchy levels at which you can apply the service filters to adaptive services interfaces:

```
[edit]
interfaces {
    interface-name {
        unit unit-number {
            family (inet | inet6) {
                service {
                    input {
                        service-set service-set-name service-filter service-filter-name;
                        post-service-filter service-filter-name;
                    }
                    output {
                        service-set service-set-name service-filter service-filter-name;
                    }
                }
            }
        }
    }
}
```

Associating Service Rules with Adaptive Services Interfaces

To define and group the service rules be applied to an adaptive services interface, you define an *interface service set* by including the `service-set service-set-name` statement at the `[edit services]` hierarchy level.

To apply an interface service set to the input and output of an adaptive services interface, you include the `service-set service-set-name` at the following hierarchy levels:

- `[edit interfaces interface-name unit unit-number input]`

- [edit interfaces *interface-name* unit *unit-number* output]

If you apply a service set to one direction of an adaptive services interface but do not apply a service set to the other direction, an error occurs when you commit the configuration.

The adaptive services PIC performs different actions depending on whether the packet is sent to the PIC for input service or for output service. For example, you can configure a single service set to perform Network Address Translation (NAT) in one direction and destination NAT (dNAT) in the other direction.

Filtering Traffic Before Accepting Packets for Service Processing

To filter IPv4 or IPv6 traffic before accepting packets for input or output service processing, include the service-set *service-set-name* service-filter *service-filter-name* at one of the following interfaces:

- [edit interfaces *interface-name* unit *unit-number* family (inet | inet6) service input]
- [edit interfaces *interface-name* unit *unit-number* family (inet | inet6) service output]

For the *service-set-name*, specify a service set configured at the [edit services service-set] hierarchy level.

The service set retains the input interface information even after services are applied, so that functions such as filter-class forwarding and destination class usage (DCU) that depend on input interface information continue to work.

The following requirements apply to filtering inbound or outbound traffic before accepting packets for service processing:

- You configure the same service set on the input and output sides of the interface.
- If you include the service-set statement without an optional service-filter definition, the Junos OS assumes the match condition is true and selects the service set for processing automatically.
- The service filter is applied only if a service set is configured and selected.

You can include more than one service set definition on each side of an interface. The following guidelines apply:

- If you include multiple service sets, the router (or switch) software evaluates them in the order in which they appear in the configuration. The system executes the first service set for which it finds a match in the service filter and ignores the subsequent definitions.
- A maximum of six service sets can be applied to an interface.
- When you apply multiple service sets to an interface, you must also configure and apply a service filter to the interface.

Postservice Filtering of Returning Service Traffic

As an option to filtering of IPv4 or IPv6 input service traffic, you can apply a service filter to IPv4 or IPv6 traffic that is returning to the services interface after the service set is executed. To apply a service filter in this manner, include the `post-service-filter` *`service-filter-name`* statement at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family (inet | inet6) service input]` hierarchy level.

RELATED DOCUMENTATION

- Service Filter Overview | **1431**
- How Service Filters Evaluate Packets | **1433**
- Guidelines for Configuring Service Filters | **1435**
- Example: Configuring and Applying Service Filters | **1441**
- *Adaptive Services and Multiservices Interfaces Overview*
- *Configuring Service Sets to be Applied to Services Interfaces*
- *Configuring Service Rules*

Example: Configuring and Applying Service Filters

IN THIS SECTION

- Requirements | **1441**
- Overview | **1442**
- Configuration | **1443**
- Verification | **1447**

This example shows how to configure and apply service filters.

Requirements

This example use the logical interface `xe-0/1/0.0` on any of the following hardware components:

- Adaptive Services (AS) PIC on an M Series or T Series router

- Multiservices (MS) PIC on an M Series or T Series router
- Multiservices (MS) DPC on an MX Series router
- EX Series switch

Before you begin, make sure that you have:

- Installed your supported router (or switch) and PICs or DPCs and performed the initial router (or switch) configuration.
- Configured basic Ethernet in the topology, and verified that traffic is flowing in the topology and that IPv4 traffic is flowing through logical interface `xe-0/1/0.0`.
- Configured the service set `vrf_svcs` with service input and output rules and default settings for services at a service interface.

For guidelines for configuring service sets, see *Configuring Service Sets to be Applied to Services Interfaces*.

Overview

IN THIS SECTION

- Topology | **1442**

In this example, you create three types of service filters for IPv4 traffic: one input service filter, one postservice input filter, and one output service filter. Different service-filters can be applied to the same service-set. See also: *Configuring Service Sets to be Applied to Services Interfaces*

Topology

You apply the input service filter and postservice input filter to input traffic at logical interface `xe-0/1/0.0`, and you apply the output service filter to the output traffic at the same logical interface.

- Filtering IPv4 traffic before it is accepted for input service processing—At logical interface `xe-0/1/0.0`, you use the service filter `in_filter_presvc` to filter IPv4 input traffic before the traffic can be accepted for processing by services associated with service set `vrf_svcs`. The `in_filter_presvc` service filter counts packets sent from ICMP port 179, directs these packets to the input services associated with the service set `vrf_svcs`, and discards all other packets.
- Filtering IPv4 traffic after it has completed input service processing—At logical interface `xe-0/1/0.0`, you use the service filter `in_filter_postsvc` to filter traffic that is returning to the services interface

after the input service set `in_filter_presvc` is executed. The `in_filter_postsvc` service filter counts packets sent from ICMP port 179 and then discards them.

- Filtering IPv4 traffic before it is accepted for output service processing—At logical interface `xe-0/1/0.0`, you use the service-filter `out_filter_presvc` to filter IPv4 output traffic before the traffic can be accepted for processing by the services associated with service set `vrf_svcs`. The `out_filter_presvc` service filter counts packets destined for TCP port 179 and then directs the packets to the output services associated with the service set `vrf_svcs`.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1443**
- Configuring the Three Service Filters | **1444**
- Applying the Three Service Filters | **1446**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet service-filter in_filter_presvc term t1 from protocol tcp
set firewall family inet service-filter in_filter_presvc term t1 from source-port bgp
set firewall family inet service-filter in_filter_presvc term t1 then count svc_in_pkts
set firewall family inet service-filter in_filter_presvc term t1 then service
set firewall family inet service-filter in_filter_postsvc term t2 from protocol tcp
set firewall family inet service-filter in_filter_postsvc term t2 from source-port bgp
set firewall family inet service-filter in_filter_postsvc term t2 then count svc_in_pkts_rtn
set firewall family inet service-filter in_filter_postsvc term t2 then skip
set firewall family inet service-filter out_filter_presvc term t3 from protocol icmp
set firewall family inet service-filter out_filter_presvc term t3 from destination-port bgp
set firewall family inet service-filter out_filter_presvc term t3 then count svc_out_pkts
set firewall family inet service-filter out_filter_presvc term t3 then service
```

```
set interfaces xe-0/1/0 unit 0 family inet service input service-set vrf_svcs service-filter
in_filter_presvc
set interfaces xe-0/1/0 unit 0 family inet service input post-service-filter in_filter_postsvc
set interfaces xe-0/1/0 unit 0 family inet service output service-set vrf_svcs service-filter
out_filter_presvc
```

Configuring the Three Service Filters

Step-by-Step Procedure

To configure the three service filters:

1. Configure the input service filter.

```
[edit]
user@host# edit firewall family inet service-filter in_filter_presvc

[edit firewall family inet service-filter in_filter_presvc]
user@host# set term t1 from protocol tcp
user@host# set term t1 from source-port bgp
user@host# set term t1 then count svc_in_pkts
user@host# set term t1 then service
```

2. Configure the postservice input filter.

```
[edit]
user@host# edit firewall family inet service-filter in_filter_postsvc

[edit firewall family inet service-filter in_filter_postsvc]
user@host# set term t2 from protocol tcp
user@host# set term t2 from source-port bgp
user@host# set term t2 then count svc_in_pkts_rtn
user@host# set term t2 then skip
```

3. Configure the output service filter.

```
[edit]
user@host# edit firewall family inet service-filter out_filter_presvc
```

```
[edit firewall family inet service-filter out_filter_presvc]
user@host# set term t3 from protocol icmp
user@host# set term t3 from destination-port bgp
user@host# set term t3 then count svc_out_pkts
user@host# set term t3 then service
```

Results

Confirm the configuration of the input and output service filters and the postservice input filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    service-filter in_filter_presvc {
        term t1 {
            from {
                protocol tcp;
                source-port bgp;
            }
            then {
                count svc_in_pkts;
                service;
            }
        }
    }
    service-filter in_filter_postsvc {
        term t2 {
            from {
                protocol tcp;
                source-port bgp;
            }
            then {
                count svc_in_pkts_rtn;
                skip;
            }
        }
    }
    service-filter out_filter_presvc {
        term t3 {
```

```
            from {
                protocol icmp;
                destination-port bgp;
            }
            then {
                count svc_out_pkts;
                service;
            }
        }
    }
}
```

Applying the Three Service Filters

Step-by-Step Procedure

To apply the three service filters:

1. Access the IPv4 protocol on the input interface `xe-0/1/0.0`.

```
[edit]
user@host# edit interfaces xe-0/1/0 unit 0 family inet
```

2. Apply the input service filter and the postservice input filter.

```
[edit interfaces xe-0/1/0 unit 0 family inet]
user@host# set service input service-set vrf_svcs service-filter in_filter_presvc
user@host# set service input post-service-filter in_filter_postsvc
user@host# set service output service-set vrf_svcs service-filter out_filter_presvc
```

Results

Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
xe-0/1/0 {
```

```
    unit 0 {
        family inet {
            service {
                input {
                    service-set vrf_svcs service-filter in_filter_presvc;
                    post-service-filter in_filter_postsvc;
                }
                output {
                    service-set vrf_svcs service-filter out_filter_presvc;
                }
            }
        }
    }
}
```

When you are done configuring the device, commit your candidate configuration.

Verification

IN THIS SECTION

- Verifying That Inbound Traffic Is Filtered Before Input Service | **1447**
- Verifying That Inbound Traffic Is Filtered After Input Service Processing | **1448**
- Verifying That Outbound Traffic Is Filtered Before Output Service Processing | **1448**

Confirm that the configuration is working properly.

Verifying That Inbound Traffic Is Filtered Before Input Service

Purpose

Verify that inbound packets sent from TCP port 179 are sent for processing by the *input* services associated with the service set `vrf_svcs`.

Action

Display the count of packets sent for processing by the *input* services associated with the service set vrf_svcs.

```
[edit]
user@host> show firewall filter in_filter_presvc-vrf_svcs counter svc_in_pkts
```

Verifying That Inbound Traffic Is Filtered After Input Service Processing

Purpose

Verify that inbound packets sent from TCP port 179 are returned from processing by the *input* services associated with the service set vrf_svcs.

Action

Display the count of packets returned from processing by the *input* services associated with the service set vrf_svcs.

```
[edit]
user@host> show firewall filter in_filter_postsvc-vrf_svcs counter svc_in_pkts_rtn
```

Verifying That Outbound Traffic Is Filtered Before Output Service Processing

Purpose

Verify that outbound packets sent to ICMP port 179 are sent for processing by the *output* services associated with the service set vrf_svcs.

Action

Display the count of packets sent for processing by the *output* services associated with the service set vrf_svcs.

```
[edit]
user@host> show firewall filter out_filter_presvc-vrf_svcs counter svc_out_pkts
```

RELATED DOCUMENTATION

Service Filter Overview | 1431

How Service Filters Evaluate Packets | 1433

Guidelines for Configuring Service Filters | 1435

Guidelines for Applying Service Filters | 1438

Service Filter Match Conditions for IPv4 or IPv6 Traffic

Service filters support only a subset of the stateless firewall filter match conditions for IPv4 and IPv6 traffic. Table 70 on page 1449 describes the service filter match conditions.

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic

Match Condition	Description	Protocol Families
`address` *address*	Match the IP source or destination address field.	• `family inet` • `family inet6`
`address` *address* `except`	Do not match the IP source or destination address field.	• `family inet` • `family inet6`
`ah-spi` *spi-value*	(M Series routers, except M120 and M320) Match on the IPsec authentication header (AH) security parameter index (SPI) value.	• `family inet`
`ah-spi-except` *spi-value*	(M Series routers, except M120 and M320) Do not match on the IPsec AH SPI value.	• `family inet`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
destination-address *address*	Match the IP destination address field. You cannot specify both the address and destination-address match conditions in the same term.	• family inet • family inet6
destination-address *address* except	Do not match the IP destination address field. You cannot specify both the address and destination-address match conditions in the same term.	• family inet • family inet6
destination-port *number*	Match the UDP or TCP destination port field. You cannot specify both the port and destination-port match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the protocol udp or protocol tcp match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the next-header udp or next-header tcp match condition in the same term to specify which protocol is being used on the port. 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), ldp (646), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nntp (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rkinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs (49), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), or xdmcp (177).	• family inet • family inet6

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`destination-port-except` *number*	Do not match the UDP or TCP destination port field. For details, see the `destination-port` match description.	• `family inet` • `family inet6`
`destination-prefix-list` *name*	Match the list of destination prefixes. The prefix list is defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level.	• `family inet` • `family inet6`
`esp-spi` *value*	Match the IPsec encapsulating security payload (ESP) SPI value. Specify a single value or a range of values. You can specify a *value* in hexadecimal, binary, or decimal form. To specify the value in hexadecimal form, include `0x` as a prefix. To specify the value in binary form, include `b` as a prefix.	• `family inet` • `family inet6`
`esp-spi-except` *value*	Do not match the IPsec ESP SPI value or range of values. For details, see the `esp-spi` match condition.	• `family inet` • `family inet6`
`first-fragment`	Match if the packet is the first fragment of a fragmented packet. Do not match if the packet is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of `0`. This match condition is an alias for the bit-field match condition `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: `first-fragment` and `is-fragment`.	• `family inet`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`forwarding-class`	Match one or more of the following specified packet forwarding classes: • `assured-forwarding` • `best-effort` • `expedited-forwarding` • `network-control` • *`user-defined-name`* For information about forwarding classes and router-internal output queues, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.	• `family inet` • `family inet6`
`forwarding-class-except`	Do not match one or more of the following specified packet forwarding classes: • `assured-forwarding` • `best-effort` • `expedited-forwarding` • `network-control` • *`user-defined-name`*	• `family inet` • `family inet6`
`fragment-flags` *`number`*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): `dont-fragment` (0x4), `more-fragments` (0x2), or `reserved` (0x8).	• `family inet`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`fragment-offset` *number*	Match the 13-bit fragment offset field in the IP header. The value is the offset, in 8-byte units, in the overall datagram message to the data fragment. Specify a numeric value, a range of values, or a set of values. An offset value of `0` indicates the first fragment of a fragmented packet. The `first-fragment` match condition is an alias for the `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`).	• `family inet`
`fragment-offset-except` *number*	Do not match the 13-bit fragment offset field.	• `family inet`
`interface-group` *group-number*	Match the interface group (set of one or more logical interfaces) on which the packet was received. For *group-number*, specify a value from `0` through `255`. For information about configuring interface groups, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 1373.	• `family inet` • `family inet6`
`interface-group-except` *group-number*	Do not match the interface group on which the packet was received. for details, see the `interface-group` match condition.	• `family inet` • `family inet6`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`ip-options` *values*	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): `loose-source-route` (131), `record-route` (7), `router-alert` (148), `security` (130), `stream-id` (136), `strict-source-route` (137), or `timestamp` (68). To match *any* value for the IP option, use the text synonym `any`. To match on *multiple* values, specify the list of values within square brackets ('[' and ']'). To match a *range* of values, use the value specification `[` *value1-value2* `]`. For example, the match condition `ip-options [ 0-147 ]` matches on an IP options field that contains the `loose-source-route`, `record-route`, or `security` values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the `router-alert` value (148). For most interfaces, a filter term that specifies an `ip-option` match on one or more *specific* IP option values (a value other than `any`) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an `ip-option` match on one or more specific IP option values, you cannot specify the `count`, `log`, or `syslog` nonterminating actions *unless* you also specify the `discard` terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router (or switch). • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the `ip-options any` match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers and EX Series switches are capable of parsing the IP option field of the IPv4 packet header. This capability is supported on EX Series switches also. For interfaces	`family inet`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
	configured on those MPCs, *all* packets that are matched using the `ip-options` match condition are sent to the Packet Forwarding Engine for processing.	
`ip-options-except` *values*	Do not match the IP option field to the specified value or list of values. For details about specifying the *values*, see the `ip-options` match condition.	• `family inet`
`is-fragment`	Match if the packet is a trailing fragment of a fragmented packet. Do not match the first fragment of a fragmented packet. This match condition is an alias for the bit-field match condition `fragment-offset 0 except` bits. **NOTE**: To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`).	• `family inet`
`loss-priority`	Match one or more of the following specified packet loss priority (PLP) levels: • `low` • `medium-low` • `medium-high` • `high` The PLP is used by schedulers in conjunction with the random early discard (RED) algorithm to control packet discard during periods of congestion. For information about PLP, see *Managing Congestion by Setting Packet Loss Priority for Different Traffic Flows* and *Overview of Assigning Service Levels to Packets Based on Multiple Packet Header Fields*.	• `family inet` • `family inet6`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`loss-priority-except`	Do not match one or more of the following specified packet loss priority (PLP) levels: • `low` • `medium-low` • `medium-high` • `high`	• `family inet` • `family inet6`
`port` *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the `destination-port` match condition or the `source-port` match condition in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the `protocol udp` or `protoco tcp` match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed under `destination-port`.	• `family inet` • `family inet6`
`port-except` *number*	Do not match the UDP or TCP source or destination port field. For details, see the `port` match condition.	• `family inet` • `family inet6`
`prefix-list` *prefix-list-name*	Match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level.	• `family inet` • `family inet6`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`protocol` *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).	• `family inet`
`protocol-except` *number*	Do not match the IP protocol type field. For details, see the `protocol` match condition.	• `family inet`
`source-address` *address*	Match the IP source address. You cannot specify both the `address` and `source-address` match conditions in the same term.	• `family inet` • `family inet6`
`source-address` *address* `except`	Do not match the IP source address. You cannot specify both the `address` and `source-address` match conditions in the same term.	• `family inet` • `family inet6`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`source-port` *number*	Match the UDP or TCP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.	• `family inet` • `family inet6`
`source-port-except` *number*	Do not match the UDP or TCP source port field. For details, see the `source-port` match condition.	• `family inet` • `family inet6`
`source-prefix-list` *name*	Match source prefixes in the specified list. Specify the name of a prefix list defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level.	• `family inet` • `family inet6`

Table 70: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(Continued)*

Match Condition	Description	Protocol Families
`tcp-flags` *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the `tcp-established` and `tcp-initial` match conditions. If you configure this match condition for IPv4 traffic, we recommend that you also configure the `protocol tcp` match statement in the same term to specify that the TCP protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the `next-header tcp` match condition in the same term to specify that the TCP protocol is being used on the port.	• `family inet` • `family inet6`

NOTE: If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see "IPv6 Overview" in the Junos OS Routing Protocols Library for Routing Devices.

RELATED DOCUMENTATION

- Service Filter Overview | 1431
- Guidelines for Configuring Service Filters | 1435
- Example: Configuring and Applying Service Filters | 1441
- Service Filter Terminating Actions | 1461
- Service Filter Nonterminating Actions | 1460

Service Filter Nonterminating Actions

Service filters support different sets of terminating actions for each protocol family.

NOTE: Service filters do not support the `next term` action.

Table 71 on page 1460 describes the nonterminating actions you can configure in a service filter term.

Table 71: Nonterminating Actions for Service Filters

Nonterminating Action	Description	Protocol Families
`count` *counter-name*	Count the packet in the named counter.	• `inet` • `inet6`
`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 at the command-line interface (CLI).	• `inet` • `inet6`
`port-mirror`	Port-mirror the packet based on the specified family. Supported on M120 routers, M320 routers configured with Enhanced III FPCs, MX Series routers, and EX Series switches only.	• `inet` • `inet6`
`sample`	Sample the packet.	• `inet` • `inet6`

RELATED DOCUMENTATION

Service Filter Overview | 1431
Guidelines for Configuring Service Filters | 1435
Example: Configuring and Applying Service Filters | 1441
Service Filter Match Conditions for IPv4 or IPv6 Traffic | 1449
Service Filter Terminating Actions | 1461

Service Filter Terminating Actions

Service filters support different sets of terminating actions than standard stateless firewall filters or simple filters.

NOTE: Service filters do not support the `next term` action.

Table 72 on page 1461 describes the terminating actions you can configure in a service filter term.

Table 72: Terminating Actions for Service Filters

Terminating Action	Description	Protocol Families
`service`	Direct the packet to service processing.	• `inet` • `inet6`
`skip`	Let the packet bypass service processing.	• `inet` • `inet6`

RELATED DOCUMENTATION

Service Filter Overview | 1431
Guidelines for Configuring Service Filters | 1435
Example: Configuring and Applying Service Filters | 1441

Service Filter Match Conditions for IPv4 or IPv6 Traffic | 1449

Service Filter Nonterminating Actions | 1460

CHAPTER 23

Configuring Simple Filters

IN THIS CHAPTER

- Simple Filter Overview | 1463
- How Simple Filters Evaluate Packets | 1464
- Guidelines for Configuring Simple Filters | 1465
- Guidelines for Applying Simple Filters | 1470
- Example: Configuring and Applying a Simple Filter | 1471

Simple Filter Overview

Simple filters are supported on Gigabit Ethernet intelligent queuing 2 (IQ2) and Enhanced Queuing Dense Port Concentrator (DPC) interfaces only.

Simple filters are recommended for metropolitan Ethernet applications.

RELATED DOCUMENTATION

How Simple Filters Evaluate Packets | 1464

Guidelines for Configuring Simple Filters | 1465

Guidelines for Applying Simple Filters | 1470

Example: Configuring and Applying a Simple Filter | 1471

How Simple Filters Evaluate Packets

IN THIS SECTION

- Simple Filters That Contain a Single Term | **1464**
- Simple Filters That Contain Multiple Terms | **1464**
- Simple Filter Terms That Do Not Contain Any Match Conditions | **1464**
- Simple Filter Terms That Do Not Contain Any Actions | **1465**
- Simple Filter Default Action | **1465**

Simple Filters That Contain a Single Term

For a simple filter that consists of a single term, the policy framework software evaluates a packet as follows:

- If the packet matches all the conditions, the actions are taken.
- If the packet matches all the conditions and no actions are specified, the packet is accepted.
- If the packet does not match all the conditions, it is discarded.

Simple Filters That Contain Multiple Terms

For a simple filter that consists of multiple terms, the policy framework software evaluates a packet against the terms in the filter sequentially, beginning with the first term in the filter, until either the packet matches all the conditions in one of the terms or there are no more terms in the filter.

- If the packet matches all the conditions in a term, the actions in that term are performed and evaluation of the packet ends at that term. Any subsequent terms in the filter are not used.
- If the packet does not match all the conditions in the term, evaluation of the packet proceeds to the next term in the filter.

Simple Filter Terms That Do Not Contain Any Match Conditions

For simple filters with a single term and for filters with multiple terms, if a term does not contain any match conditions, the actions are taken on any packet evaluated.

Simple Filter Terms That Do Not Contain Any Actions

If a simple filter term does not contain any actions, and if the packet matches the conditions in the term, the packet is accepted.

Simple Filter Default Action

Each simple filter has an *implicit* `discard` action at the end of the filter, which is equivalent to including the following example term `explicit_discard` as the final term in the simple filter:

```
term explicit_discard {
    then discard;
}
```

By default, if a packet matches none of the terms in a simple filter, the packet is discarded.

RELATED DOCUMENTATION

- Simple Filter Overview | **1463**
- Guidelines for Configuring Simple Filters | **1465**
- Guidelines for Applying Simple Filters | **1470**
- Example: Configuring and Applying a Simple Filter | **1471**

Guidelines for Configuring Simple Filters

IN THIS SECTION

- Statement Hierarchy for Configuring Simple Filters | **1466**
- Simple Filter Protocol Families | **1466**
- Simple Filter Names | **1466**
- Simple Filter Terms | **1466**
- Simple Filter Match Conditions | **1467**
- Simple Filter Terminating Actions | **1469**
- Simple Filter Nonterminating Actions | **1469**

Statement Hierarchy for Configuring Simple Filters

To configure a simple filter, include the `simple-filter` *`simple-filter-name`* statement at the `[edit firewall family inet]` hierarchy level.

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

Individual statements supported under the `simple-filter` *`simple-filter-name`* statement are described separately in this topic and are illustrated in the example of configuring and applying a simple filter.

Simple Filter Protocol Families

You can configure simple filters to filter IPv4 traffic (`family inet`) only. No other protocol family is supported for simple filters.

Simple Filter Names

Under the `family inet` statement, you can include `simple-filter` *`simple-filter-name`* statements to create and name simple filters. 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 (“ ”).

Simple Filter Terms

Under the `simple-filter` *`simple-filter-name`* statement, you can include `term` *`term-name`* statements to create and name filter terms.

- You must configure at least one term in a *firewall filter*.

- You must specify a unique name for each term within a firewall filter. The term name 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 (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter 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` configuration mode command to reorder the terms of a firewall filter.

Simple filters do *not* support the `next term` action.

Simple Filter Match Conditions

Simple filter terms support only a subset of the IPv4 match conditions that are supported for standard stateless firewall filters.

Unlike standard stateless firewall filters, the following restrictions apply to simple filters:

- On MX Series routers with the Enhanced Queuing DPC and on EX Series switches, simple filters do *not* support the `forwarding- class` match condition.
- Simple filters support only one `source-address` and one `destination-address` prefix for each filter term. If you configure multiple prefixes, only the last one is used.
- Simple filters do *not* support multiple source addresses and destination addresses in a single term. If you configure multiple addresses, only the last one is used.
- Simple filters do *not* support negated match conditions, such as the `protocol-except` match condition or the `exception` keyword.
- Simple filters support a range of values for `source-port` and `destination-port` match conditions only. For example, you can configure `source-port 400-500` or `destination-port 600-700`.
- Simple filters do *not* support noncontiguous mask values.

Table 73 on page 1467 lists the simple filter match conditions.

Table 73: Simple Filter Match Conditions

Match Condition	Description
`destination-address` *`destination-address`*	Match IP destination address.

Table 73: Simple Filter Match Conditions *(Continued)*

Match Condition	Description
`destination-port` *number*	TCP or UDP destination port field. If you configure this match condition, we recommend that you also configure the `protocol` match statement to determine which protocol is being used on the port. In place of the numeric value, you can specify one of the following text aliases (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), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (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), or `xdmcp` (177).
`forwarding-class` *class*	Match the forwarding class of the packet. Specify `assured-forwarding`, `best-effort`, `expedited-forwarding`, or `network-control`. For information about forwarding classes and router-internal output queues, see *Understanding How Forwarding Classes Assign Classes to Output Queues*.
`protocol` *number*	IP protocol field. In place of the numeric value, you can specify one of the following text aliases (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).
`source-address` *ip-source-address*	Match the IP source address.

Table 73: Simple Filter Match Conditions *(Continued)*

Match Condition	Description
`source-port` *number*	Match the UDP or TCP source port field. If you configure this match condition, we recommend that you also configure the `protocol` match statement to determine which protocol is being used on the port. In place of the numeric field, you can specify one of the text aliases listed for `destination-port`.

Simple Filter Terminating Actions

Simple filters do *not* support explicitly configurable terminating actions, such as `accept`, `reject`, and `discard`. Terms configured in a simple filter always accept packets.

Simple filters do *not* support the `next` action.

Simple Filter Nonterminating Actions

Simple filters support only the following nonterminating actions:

- `forwarding-class (`*forwarding-class*` | assured-forwarding |best-effort | expedited-forwarding | network-control)`

  NOTE: On the MX Series routers and EX Series switches with the Enhanced Queuing DPC, the forwarding class is not supported as a `from` match condition.

- `loss-priority (high | low | medium-high | medium-low)`

Simple filters do not support actions that 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).

RELATED DOCUMENTATION

Simple Filter Overview | 1463

How Simple Filters Evaluate Packets | 1464

Guidelines for Applying Simple Filters | 1470

Example: Configuring and Applying a Simple Filter | 1471

Guidelines for Applying Simple Filters

IN THIS SECTION

- Statement Hierarchy for Applying Simple Filters | 1470
- Restrictions for Applying Simple Filters | 1470

Statement Hierarchy for Applying Simple Filters

You can apply a simple filter to the IPv4 ingress traffic at a *logical interface* by including the `simple-filter input` *`simple-filter-name`* statement at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet]` hierarchy level.

```
[edit]
interfaces {
    interface-name {
        unit logical-unit-number {
            family inet {
                simple-filter {
                    input filter-name;
                }
            }
        }
    }
}
```

Restrictions for Applying Simple Filters

You can apply a simple filter to the ingress IPv4 traffic at a logical interface configured on the following hardware only:

- Gigabit Ethernet intelligent queuing (IQ2) PICs installed on M120, M320, or T Series routers.

- Enhanced Queuing Dense Port Concentrators (EQ DPCs) installed on MX Series routers and EX Series switches.

The following additional restrictions pertain to applying simple filters:

- Simple filters are not supported on Modular Port Concentrator (MPC) interfaces, including Enhanced Queuing MPC interfaces.
- Simple filters are not supported for interfaces in an aggregated-Ethernet bundle.
- You can apply simple filters to `family inet` traffic only. No other protocol family is supported.
- You can apply simple filters to ingress traffic only. Egress traffic is not supported.
- You can apply only a single simple filter to a supported logical interface. Input lists are not supported.

RELATED DOCUMENTATION

Simple Filter Overview | 1463
How Simple Filters Evaluate Packets | 1464
Guidelines for Configuring Simple Filters | 1465
Example: Configuring and Applying a Simple Filter | 1471

Example: Configuring and Applying a Simple Filter

IN THIS SECTION

- Requirements | 1471
- Overview | 1472
- Configuration | 1472
- Verification | 1476

This example shows how to configure a simple filter.

Requirements

This example uses one of the following hardware components:

- One Gigabit Ethernet intelligent queuing (IQ2) PIC installed on an M120, M320, or T Series router
- One Enhanced Queuing Dense Port Concentrator (EQ DPC) installed on an MX Series router or an EX Series switch

Before you begin, make sure that you have:

- Installed your supported router (or switch) and PIC or DPC and performed the initial router (or switch) configuration.
- Configured basic Ethernet in the topology, and verified that traffic is flowing in the topology and that ingress IPv4 traffic is flowing into logical interface `ge-0/0/1.0`.

Overview

IN THIS SECTION

- Topology | **1472**

This simple filter sets the loss priority to low for TCP traffic with source address `172.16.1.1`, sets the loss priority to high for HTTP (Web) traffic with source addresses in the 172.16.4.0/8 range, and sets the loss priority to low for all traffic with destination address `172.16.6.6`.

Topology

The simple filter is applied as an input filter (arriving packets are checking for destination address `6.6.6.6`, not queued output packets) on interface `ge-0/0/1.0`.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1473**
- Configuring the Simple Firewall Filter | **1473**
- Applying the Simple Filter to the Logical Interface Input | **1475**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall family inet simple-filter sf_classify_1 term 1 from source-address 172.16.1.1/32
set firewall family inet simple-filter sf_classify_1 term 1 from protocol tcp
set firewall family inet simple-filter sf_classify_1 term 1 then loss-priority low
set firewall family inet simple-filter sf_classify_1 term 2 from source-address 172.16.4.0/8
set firewall family inet simple-filter sf_classify_1 term 2 from protocol tcp
set firewall family inet simple-filter sf_classify_1 term 2 from source-port http
set firewall family inet simple-filter sf_classify_1 term 2 then loss-priority high
set firewall family inet simple-filter sf_classify_1 term 3 from destination-address 6.6.6.6/32
set firewall family inet simple-filter sf_classify_1 term 3 then loss-priority low
set firewall family inet simple-filter sf_classify_1 term 3 then forwarding-class best-effort
set interfaces ge-0/0/1 unit 0 family inet simple-filter input sf_classify_1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configuring the Simple Firewall Filter

Step-by-Step Procedure

To configure the simple filter:

1. Create the simple filter `sf_classify_1`.

```
[edit]
user@host# edit firewall family inet simple-filter sf_classify_1
```

2. Configure classification of TCP traffic based on the source IP address.

```
[edit firewall family inet simple-filter sf_classify_1]
user@host# set term 1 from source-address 172.16.1.1/32
user@host# set term 1 from protocol tcp
user@host# set term 1 then loss-priority low
```

3. Configure classification of HTTP traffic based on the source IP address.

```
[edit firewall family inet simple-filter sf_classify_1]
user@host# set term 2 from source-address 172.16.4.0/8
user@host# set term 2 from protocol tcp
user@host# set term 2 from source-port http
user@host# set term 2 then loss-priority high
```

4. Configure classification of other traffic based on the destination IP address.

```
[edit firewall family inet simple-filter sf_classify_1]
user@host# set term 3 from destination-address 6.6.6.6/32
user@host# set term 3 then loss-priority low
user@host# set term 3 then forwarding-class best-effort
```

Results

Confirm the configuration of the simple filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    simple-filter sf_classify_1 {
        term 1 {
            from {
                source-address {
                    172.16.1.1/32;
                }
                protocol {
                    tcp;
                }
            }
            then loss-priority low;
        }
        term 2 {
            from {
                source-address {
                    172.16.4.0/8;
```

```
                }
                source-port {
                    http;
                }
                protocol {
                    tcp;
                }
            }
            then loss-priority high;
        }
        term 3 {
            from {
                destination-address {
                    6.6.6.6/32;
                }
            }
            then {
                loss-priority low;
                forwarding-class best-effort;
            }
        }
    }
}
```

Applying the Simple Filter to the Logical Interface Input

Step-by-Step Procedure

To apply the simple filter to the logical interface input:

1. Configure the logical interface to which you will apply the simple filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the simple filter to the logical interface input.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set simple-filter input sf_classify_1
```

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            simple-filter {
                input sf_classify_1;
            }
            address 10.1.2.3/30;
        }
    }
}
```

When you are done configuring the device, commit your candidate configuration.

Verification

IN THIS SECTION

- Displaying the Mapping of Forwarding Class Maps and Names to Queue Numbers | **1477**
- Displaying CoS Queue Counters for the Interface | **1477**
- Displaying CoS Queue Counter Details for the Physical Interface | **1478**

Confirm that the configuration is working properly.

Displaying the Mapping of Forwarding Class Maps and Names to Queue Numbers

Purpose

Display the mapping of forwarding class names to queue numbers.

Action

Enter the `show class-of-service forwarding-class` operational mode command.

```
[edit]
user@host> show class-of-service forwarding-class
```

For information about the command output, see "`show class-of-service forwarding-class`" in the CLI Explorer.

Displaying CoS Queue Counters for the Interface

Purpose

Verify that the class-of-service (CoS) queue counters for the interface reflect the simple filter applied to the logical interface.

Action

Enter the `show interfaces` command for the physical interface on which the simple filter is applied, and specify `detail` or `extensive` output level.

```
[edit]
user@host> show interfaces ge-0/0/1 detail
```

In the `Physical interface` section, under `Ingress queues`, the `Queue counters` section displays ingress queue counters for each forwarding class.

For more detailed information about the command output, see "`show interfaces` " in the CLI Explorer.

Displaying CoS Queue Counter Details for the Physical Interface

Purpose

Verify that the CoS queue counter details for the physical interface reflect the simple filter applied to the logical interface.

Action

Enter the `show interfaces queue` command for the physical interface on which the simple filter is applied, and specify the `ingress` option.

```
[edit]
user@host> show interfaces queue ge-0/0/1 ingress
```

For information about the command output, see "`show interfaces queue`" in the CLI Explorer.

RELATED DOCUMENTATION

- Simple Filter Overview | 1463
- How Simple Filters Evaluate Packets | 1464
- Guidelines for Configuring Simple Filters | 1465
- Guidelines for Applying Simple Filters | 1470

CHAPTER 24

Configuring Layer 2 Firewall Filters

IN THIS CHAPTER

- Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | **1479**
- Example: Configuring Filtering of Frames by MAC Address | **1480**
- Example: Configuring Filtering of Frames by IEEE 802.1p Bits | **1482**
- Example: Configuring Filtering of Frames by Packet Loss Priority | **1483**
- Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | **1485**
- Understanding Firewall Filters on OVSDB-Managed Interfaces | **1488**
- Example: Applying a Firewall Filter to OVSDB-Managed Interfaces | **1489**

Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances

Juniper Networks MX Series 5G Universal Routing Platforms support firewall filters for the `bridge` and `vpls` protocol families. You configure these firewall filters to control traffic within bridge domains and VPLS instances. This topic explores some of the ways that filters can be used in a Layer 2 environment to control traffic.

MX Series router firewall filters can be applied to:

- Input interfaces
- Output interfaces
- Input to the Layer 2 forwarding table

You use a *firewall filter* after taking the following two steps:

1. You configure any policers and the firewall filter at the `[edit firewall]` hierarchy level.
2. You apply the properly configured firewall filter to an interface or bridge domain.

NOTE: If the chassis is running in Enhanced IP mode, a single shared filter instance is created for a filter applied across bridge domains. Otherwise, separate filter instances are created for each bridge domain that the filter is applied to.

RELATED DOCUMENTATION

Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | 1485
Example: Configuring Filtering of Frames by MAC Address | 1480
Example: Configuring Filtering of Frames by IEEE 802.1p Bits | 1482
Example: Configuring Filtering of Frames by Packet Loss Priority | 1483

Example: Configuring Filtering of Frames by MAC Address

This example firewall filter finds frames with a certain source MAC address (**88:05:00:29:3c:de/48**), then counts and silently discards them. For more information about configuring firewall filter match conditions, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide. The filter is applied to the VLAN configured as **vlan100200** as an input filter on Router 1.

NOTE: This example does not present exhaustive configuration listings for all routers in the figures. However, you can use this example with a broader configuration strategy to complete the MX Series router network Ethernet Operations, Administration, and Maintenance (OAM) configurations.

To configure filtering of frames by MAC address:

1. Configure **evil-mac-address**, the firewall filter:

```
[edit firewall]
family bridge {
    filter evil-mac-address {
        term one {
            from {
                source-mac-address 88:05:00:29:3c:de/48;
            }
```

```
            then {
                count evil-mac-address; # Counts frame with the bad source MAC address
                discard;
            }
            term two {
                then accept; # Make sure to accept other traffic
            }
        }
    }
}
```

2. Apply **evil-mac-address** as an input filter to **vlan100200** on Router 1:

```
[edit routing-instances]
virtual-switch-R1-1 {
    bridge-domains {
        vlan100200 {
            domain-type bridge;
            forwarding-options {
                filter {
                    input evil-mac-address;
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | **1479**

Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | **1485**

Example: Configuring Filtering of Frames by IEEE 802.1p Bits | **1482**

Example: Configuring Filtering of Frames by Packet Loss Priority | **1483**

Example: Configuring Filtering of Frames by IEEE 802.1p Bits

For the **bridge** and **vpls** protocol families only, MX Series router firewall filters can be configured to provide matching on IEEE 802.1p priority bits in packets with VLAN tagging:

- To configure a firewall filter term that includes matching on IEEE 802.1p learned VLAN priority (in the outer VLAN tag), use the **learn-vlan-1p-priority** or **learn-vlan-1p-priority-except** match condition.
- To configure a firewall filter term that includes matching on IEEE 802.1p user priority (in the inner VLAN tag), use the **user-vlan-1p-priority** or **user-vlan-1p-priority-except** match condition.

For more detailed information about configuring firewall filters and configuring filter match conditions for Layer 2 bridging traffic on the MX Series routers, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

> **NOTE**: Layer 2 bridging is supported only on the MX Series routers. For more information about how to configure Layer 2 bridging, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

This example Layer 2 bridging firewall filter finds any incoming frames with an IEEE 802.1p learned VLAN priority level of either 1 or 2, and then classifies the packet in the **best-effort** default forwarding class.

> **NOTE**: This example does not present exhaustive configuration listings for all routers in the figures. However, you can use this example with a broader configuration strategy to complete the MX Series router network Ethernet Operations, Administration, and Maintenance (OAM) configurations.

To configure filtering of frames by IEEE 802.1p bits:

1. Configure the firewall filter **filter-learn-vlan-configure-forwarding**:

```
[edit firewall]
family bridge {
    filter filter-learn-vlan-configure-forwarding {
        term 0 {
            from {
                learn-vlan-1p-priority [1 2];
            }
            then forwarding-class best-effort;
```

```
        }
    }
}
```

2. Apply the firewall filter **filter-learn-vlan-configure-forwarding** as an input filter to **ge-0/0/0**:

```
[edit interfaces]
ge-0/0/0 {
    unit 0 {
        family bridge {
            filter {
                input filter-learn-vlan-configure-forwarding;
            }
        }
    }
}
```

RELATED DOCUMENTATION

- Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | 1479
- Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | 1485
- Example: Configuring Filtering of Frames by MAC Address | 1480
- Example: Configuring Filtering of Frames by Packet Loss Priority | 1483

Example: Configuring Filtering of Frames by Packet Loss Priority

To configure an MX Series router firewall filter to provide matching on the packet loss priority (PLP) level carried in the frame, use the **loss-priority** or **loss-priority-except** match condition. Packet loss priority matching is available for all protocols. For more detailed information about configuring firewall filters and configuring filter match conditions for Layer 2 bridging traffic on the MX Series routers, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

NOTE: Layer 2 bridging is supported only on the MX Series routers. For more information about how to configure Layer 2 bridging, see the Junos OS Routing Protocols Library for Routing Devices.

This example Layer 2 bridging firewall filter finds any incoming frames with a packet loss priority (PLP) level of **medium-high**, and then classifies the packet in the **expedited-forwarding** default forwarding class.

NOTE: This example does not present exhaustive configuration listings for all routers in the figures. However, you can use this example with a broader configuration strategy to complete the MX Series router network Ethernet Operations, Administration, and Maintenance (OAM) configurations.

To configure filtering of frames by packet loss priority:

1. Configure the firewall filter **filter-plp-configure-forwarding**:

```
[edit firewall]
family bridge {
    filter filter-plp-configure-forwarding {
        term 0 {
            from {
                loss-priority medium-high;
            }
            then forwarding-class expedited-forwarding;
        }
    }
}
```

2. Configure a Layer 2 bridging domain **bd** for the **ge-0/0/0** interface (that has already been configured at the `[edit interfaces]` hierarchy level):

```
[edit bridge-domains]
bd {
    domain-type bridge {
        interface ge-0/0/0;
```

```
    }
}
```

3. Apply the filter **filter-plp-configure-forwarding** as an input filter to the **ge-0/0/0** interface:

```
[edit interfaces]
ge-0/0/0 {
    unit 0 {
        family bridge {
            filter {
                input filter-plp-configure-forwarding;
            }
        }
    }
}
```

RELATED DOCUMENTATION

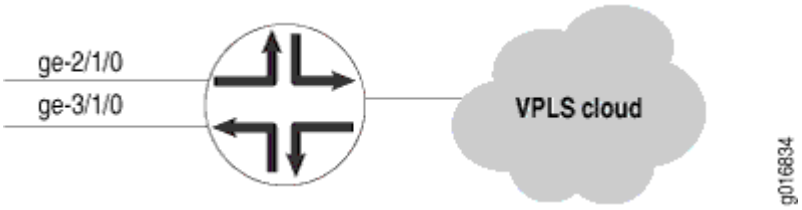
- Routing Policies, Firewall Filters, and Traffic Policers User Guide
- Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | **1479**
- Example: Configuring Policing and Marking of Traffic Entering a VPLS Core | **1485**
- Example: Configuring Filtering of Frames by MAC Address | **1480**
- Example: Configuring Filtering of Frames by IEEE 802.1p Bits | **1482**

Example: Configuring Policing and Marking of Traffic Entering a VPLS Core

This example firewall filter allows a service provider to limit the aggregate broadcast traffic entering the virtual private LAN service (VPLS) core. The broadcast, unknown unicast, and non-IP multicast traffic received from one of the service provider's customers on a logical interface has a policer applied. The service provider has also configured a two-rate, three-color policer to limit the customer's IP multicast traffic. For more information on the configuration of policers, see the Junos OS Class of Service User Guide for Routing Devices.

The position of the router is shown in Figure 65 on page 1486.

Figure 65: Policing and Marking Traffic Entering a VPLS Core



There are four major parts to the configuration:

- The policer for broadcast, unknown unicast, and non-IP multicast traffic. This example marks the loss priority as high if this type of traffic exceeds 50 Kbps.
- The two-rate, three-color policer for IP multicast traffic. This example configures a committed information rate (CIR) of 4 Mbps, a committed burst size of 256 Kbytes, a peak information rate of 4.1 Mbps, and a peak burst size of 256 Kbytes (the same as the CIR).
- The filter that applies the two policers to VPLS.
- The application of the filter to the customer interface configuration as an input filter.

NOTE: This example does not present exhaustive configuration listings for all routers in the figures. However, you can use this example with a broader configuration strategy to complete the MX Series router network Ethernet Operations, Administration, and Maintenance (OAM) configurations.

To configure policing and marking of traffic entering a VPLS core:

1. Configure **policer bcast-unknown-unicast-non-ip-mcast-policer**, a firewall policer to limit the aggregate broadcast, unknown unicast, and non-IP multicast to 50 kbps:

```
[edit firewall]
policer bcast-unknown-unicast-non-ip-mcast-policer {
    if-exceeding {
        bandwidth-limit 50k;
        burst-size-limit 150k;
    }
    then loss-priority high;
}
```

2. Configure **three-color-policer ip-multicast-traffic-policer**, a three-color policer to limit the IP multicast traffic:

```
[edit firewall]
three-color-policer ip-multicast-traffic-policer {
    two-rate {
        color-blind;
        committed-information-rate 4m;
        committed-burst-size 256k;
        peak-information-rate 4100000;
        peak-burst-size 256k;
    }
}
```

3. Configure **customer-1**, a firewall filter that uses the two policers to limit and mark customer traffic. The first term marks the IP multicast traffic based on the destination MAC address, and the second term polices the broadcast, unknown unicast, and non-IP multicast traffic:

```
[edit firewall]
family vpls {
    filter customer-1 {
        term t0 {
            from {
                destination-mac-address {
                    01:00:5e:00:00:00/24;
                }
            }
            then {
                three-color-policer {
                    two-rate ip-multicast-traffic-policer;
                }
                forwarding-class expedited-forwarding;
            }
        }
        term t1 {
            from {
                traffic-type [ broadcast unknown-unicast multicast ];
            }
            then policer bcast-unknown-unicast-non-ip-mcast-policer;
        }
```

```
    }
}
```

4. Apply the firewall filter as an input filter to the customer interface at **ge-2/1/0**:

```
[edit interfaces]
ge-2/1/0 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 5 {
        encapsulation vlan-vpls;
        vlan-id 9;
        family vpls {
            filter {
                input customer-1;
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding Firewall Filters Used to Control Traffic Within Bridge Domains and VPLS Instances | **1479**

Example: Configuring Filtering of Frames by MAC Address | **1480**

Example: Configuring Filtering of Frames by IEEE 802.1p Bits | **1482**

Example: Configuring Filtering of Frames by Packet Loss Priority | **1483**

Understanding Firewall Filters on OVSDB-Managed Interfaces

When you use a Contrail controller to manage VXLANs on a QFX switch (through the Open vSwitch Database—OVSDB—management protocol), the VXLAN interfaces are automatically configured with the `flexible-vlan-tagging` and `encapsulation extended-vlan-bridge` statements. Starting with Junos OS Release 14.1X53-D30, you can create `family ethernet-switching` logical units (subinterfaces) on these interfaces. This enables you to apply Layer 2 (`family ethernet-switching`) firewall filters to these subinterfaces, which means that you apply firewall filters to OVSDB-managed interfaces. These filters support all the same match conditions and actions as any other Layer 2 filter.

WARNING: Firewall filters are the only supported configuration items on `family ethernet-switching` subinterfaces of OVSDB-managed interfaces. Layer 2 (port) filters are the only allowed filters.

Because a Contrail controller can create subinterfaces dynamically, you need to apply firewall filters in such a way that the filters will apply to subinterfaces whenever the controller creates them. You accomplish this by using configuration groups to configure and apply the firewall filters. See "Example: Applying a Firewall Filter to OVSDB-Managed Interfaces" on page 1489 for more information.

RELATED DOCUMENTATION

Example: Applying a Firewall Filter to OVSDB-Managed Interfaces | **1489**

Overview of Firewall Filters (QFX Series) | **1716**

Understanding VXLANs

Understanding the OVSDB Protocol Running on Juniper Networks Devices

Understanding Policers on OVSDB-Managed Interfaces | **2028**

Example: Applying a Firewall Filter to OVSDB-Managed Interfaces

IN THIS SECTION

- Requirements | **1490**
- Overview | **1490**
- Configuration | **1490**

Starting with Junos OS Release 14.1X53-D30, you can create `family ethernet-switching` logical units (subinterfaces) on VXLAN interfaces managed by a Contrail controller. (The controller and switch communicate through the Open vSwitch Database—OVSDB—management protocol). This support enables you to apply Layer 2 (`family ethernet-switching`) firewall filters to these subinterfaces, which means that you apply firewall filters to OVSDB-managed interfaces. Because a Contrail controller can create subinterfaces dynamically, you need to apply firewall filters in such a way that the filters will apply to subinterfaces whenever the controller creates them. You accomplish this by using configuration groups

to configure and apply the firewall filters. (You must use configuration groups for this purpose—that is, you cannot apply a firewall filter directly to these subinterfaces.)

NOTE: Firewall filters are the only supported configuration items on `family ethernet-switching` subinterfaces of OVSDB-managed interfaces. Layer 2 (port) filters are the only allowed filters.

Requirements

This example uses the following hardware and software components:

- A QFX5100 switch
- Junos OS Release 14.1X53-D30 or later

Overview

This example assumes that interfaces xe-0/0/0 and xe-0/0/1 on the switch are VXLAN interfaces managed by a Contrail controller, which means that the controller has applied the `flexible-vlan-tagging` and `encapsulation extended-vlan-bridge` statements to these interfaces. You want to apply a firewall filter that accepts traffic from the Web to any subinterfaces that the controller creates dynamically. To apply a firewall filter Layer 2 (port) firewall filter to any dynamically created subinterfaces, you must create and apply the filter as shown in this example.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1490**
- Procedure | **1491**

To configure a firewall filter to be automatically applied to subinterfaces created dynamically by a Contrail controller, perform these tasks:

CLI Quick Configuration

```
[edit]
set groups vxlan-filter-group interfaces xe-0/0/0 unit <*> family ethernet-switching filter
input vxlan-filter
set groups vxlan-filter-group interfaces xe-0/0/1 unit <*> family ethernet-switching filter
```

```
input vxlan-filter
set groups vxlan-filter-group firewall family ethernet-switching filter vxlan-filter term t1
from destination-port 80
set groups vxlan-filter-group firewall family ethernet-switching filter vxlan-filter term t1
then accept
set apply-groups vxlan-filter-group
```

Procedure

Step-by-Step Procedure

1. Create configuration group `vxlan-filter-group` to apply firewall filter `vxlan-filter` to any subinterface of interface xe-0/0/0. The filter applies to any subinterface because you specify `unit <*>`:

   ```
   [edit]
   user@switch# set groups vxlan-filter-group interfaces xe-0/0/0 unit <*> family ethernet-
   switching filter input vxlan-filter
   ```

2. Create the same configuration for interface xe-0/0/1:

   ```
   [edit]
   user@switch# set groups vxlan-filter-group interfaces xe-0/0/1 unit <*> family ethernet-
   switching filter input vxlan-filter
   ```

3. Configure the group to include a family `ethernet-switching` filter that matches on outgoing traffic to the web:

   ```
   [edit]
   user@switch# set groups vxlan-filter-group firewall family ethernet-switching filter vxlan-
   filter term t1 from destination-port 80
   ```

4. Configure the group to accept the traffic that matches the filter:

   ```
   [edit]
   user@switch# set groups vxlan-filter-group firewall family ethernet-switching filter vxlan-
   filter term t1 then accept
   ```

5. Apply the group to enable its configuration:

```
[edit]
user@switch# set apply-groups vxlan-filter-group
```

RELATED DOCUMENTATION

- *Understanding Junos OS Configuration Groups*
- Overview of Firewall Filters (QFX Series) | **1716**
- *Understanding VXLANs*
- *Understanding the OVSDB Protocol Running on Juniper Networks Devices*
- Example: Applying a Policer to OVSDB-Managed Interfaces | **2029**

CHAPTER 25

Configuring Firewall Filters for Forwarding, Fragments, and Policing

IN THIS CHAPTER

- Filter-Based Forwarding Overview | **1493**
- Firewall Filters That Handle Fragmented Packets Overview | **1496**
- Stateless Firewall Filters That Reference Policers Overview | **1496**
- Example: Configuring Filter-Based Forwarding on the Source Address | **1497**
- Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**

Filter-Based Forwarding Overview

IN THIS SECTION

- Filters That Classify Packets or Direct Them to Routing Instances | **1494**
- Input Filtering to Classify and Forward Packets Within the Router or Switch | **1495**
- Output Filtering to Forward Packets to Another Routing Table | **1495**
- Restrictions for Applying Filter-Based Forwarding | **1495**

Firewall filters can be used to block specific packets. They can also be used to affect how specific packets are forwarded.

Filters That Classify Packets or Direct Them to Routing Instances

For IPv4 or IPv6 traffic only, you can use stateless firewall filters in conjunction with forwarding classes and routing instances to control how packets travel in a network. This is called *filter-based forwarding* (FBF).

You can define a filtering term that matches incoming packets based on source address and then classifies matching packets to a specified forwarding class. This type of filtering can be configured to grant certain types of traffic preferential treatment or to improve load balancing. To configure a stateless *firewall filter* to classify packets to a forwarding class, configure a term with the *nonterminating action* `forwarding-class` *`class-name`*.

You can also define a filtering term that directs matching packets to a specified routing instance. This type of filtering can be configured to route specific types of traffic through a firewall or other security device before the traffic continues on its path. To configure a stateless firewall filter to direct traffic to a routing instance, configure a term with the *terminating action* `routing-instance` *`routing-instance-name`* `<topology` *`topology-name`*`>` to specify the routing instance to which matching packets will be forwarded.

NOTE: Unicast Reverse Path Forwarding (uRPF) check is compatible with FBF actions. uRPF check is processed for source address checking before any FBF actions are enabled for static and dynamic interfaces. This applies to both IPv4 and IPv6 families.

NOTE: Services Offload (SOF) and Power Mode Ipsec (PMI) path will not be followed by the packets which are forwarded with FBF.

- SOF - Even if SOF is enabled , the packets will not go through SOF if they are forwarded with FBF.
- PMI - If PMI is configured, the direction to which the filter is configured, the packets in that direction will not go through the PMI. The returning packets will go through the PMI, provided the returning packets are not forwarded with FBF.

To forward traffic to the master routing instance, reference `routing-instance default` in the firewall configuration, as shown here:

```
[edit firewall]
family inet {
    filter test {
        term 1 {
            then {
```

```
                routing-instance default;
            }
        }
    }
}
```

NOTE: Do not reference `routing-instance master`. This does not work.

Input Filtering to Classify and Forward Packets Within the Router or Switch

You can configure filters to classify packets based on source address and specify the forwarding path the packets take within the router or switch 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 or switch 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.

Output Filtering to Forward Packets to Another Routing Table

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.

Restrictions for Applying Filter-Based Forwarding

An interface configured with filter-based forwarding does not support source-class usage (SCU) filter matching or source-class and destination-class usage (SCU/DCU) accounting.

If filter-based forwarding is directly attached to an interface or through forwarding table filter, then the DCU on that interface will not work.

RELATED DOCUMENTATION

Example: Configuring Filter-Based Forwarding on the Source Address | 1497

Example: Configuring Filter-Based Forwarding on Logical Systems | 1236

Firewall Filters That Handle Fragmented Packets Overview

You can create stateless firewall filters that handle fragmented packets destined for the Routing Engine. By applying these policies to the Routing Engine, you protect against the use of IP fragmentation as a means to disguise TCP packets from a *firewall filter*.

For example, consider an IP packet that is fragmented into the smallest allowable fragment size of 8 bytes (a 20-byte IP header plus an 8-byte payload). If this IP packet carries a TCP packet, the first fragment (fragment offset of 0) that arrives at the device contains only the TCP source and destination ports (first 4 bytes), and the sequence number (next 4 bytes). The TCP flags, which are contained in the next 8 bytes of the TCP header, arrive in the second fragment (fragment offset of 1).

See RFC 1858, *Security Considerations for IP Fragment Filtering*.

RELATED DOCUMENTATION

Understanding How to Use Standard Firewall Filters | 778

Example: Configuring a Stateless Firewall Filter to Handle Fragments | 1193

Stateless Firewall Filters That Reference Policers Overview

Policing, or rate limiting, is an important component of firewall filters that lets you limit the amount of traffic that passes into or out of an interface.

A *firewall filter* that references a policer can provide protection from denial-of-service (DOS) attacks. Traffic that exceeds the rate limits configured for the policer is either discarded or marked as lower priority than traffic that conforms to the configured rate limits. Packets can be marked for a lower priority by being set to a specific output queue, set to a specific packet loss priority (PLP) level, or both. When necessary, low-priority traffic can be discarded to prevent congestion.

A policer specifies two types of rate limits on traffic:

- Bandwidth limit—The average traffic rate permitted, specified as a number of bits per second.
- Maximum burst size—The packet size permitted for bursts of data that exceed the bandwidth limit.

Policing uses an algorithm to enforce a limit on average bandwidth while allowing bursts up to a specified maximum value. You can use policing to define specific classes of traffic on an interface and apply a set of rate limits to each class. After you name and configure a policer, it is stored as a template. You can then apply the policer in an interface configuration or, to rate-limit packet-filtered traffic only, in a firewall filter configuration.

For an IPv4 firewall filter term only, you can also specify a *prefix-specific action* as a nonterminating action that applies a policer to the matched packets. A prefix-specific action applies additional matching criteria on the filter-matched packets based on specified address prefix bits and then associates the matched packets with a counter and policer instance for that filter term or for all terms in the firewall filter.

To apply a policer or a prefix action to packet-filtered traffic, you can use the following firewall filter nonterminating actions:

- `policer` *`policer-name`*
- `three-color-policer (single-rate | two-rate)` *`policer-name`*
- `prefix-action` *`action-name`*

NOTE: The packet lengths that a policer considers depends on the address family of the firewall filter. See "Understanding the Frame Length for Policing Packets" on page 1926.

RELATED DOCUMENTATION

Firewall Filter Nonterminating Actions | 871

Controlling Network Access Using Traffic Policing Overview | 1916

Prefix-Specific Counting and Policing Actions | 2083

Example: Configuring Filter-Based Forwarding on the Source Address

IN THIS SECTION

Requirements | 1498

Overview | 1498

Configuration | 1501
Verification | 1509

This example shows how to configure filter-based forwarding (FBF), which is sometimes also called Policy Based Routing (PBR). The filter classifies packets to determine their forwarding path within the ingress routing device.

Filter-based forwarding is supported for IP version 4 (IPv4) and IP version 6 (IPv6).

Requirements

No special configuration beyond device initialization is required for this example.

Overview

IN THIS SECTION

Topology | 1500

In this example, we use FBF for service provider selection when customers have Internet connectivity provided by different ISPs yet share a common access layer. When a shared media (such as a cable modem) is used, a mechanism on the common access layer looks at Layer 2 or Layer 3 addresses and distinguishes between customers. You can use filter-based forwarding when the common access layer is implemented using a combination of Layer 2 switches and a single router.

With FBF, all packets received on an interface are considered. Each packet passes through a filter that has match conditions. If the match conditions are met for a filter and you have created a routing instance, FBF is applied to the packet. The packet is forwarded based on the next hop specified in the routing instance. For static routes, the next hop can be a specific LSP.

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

To configure FBF, perform the following tasks:

- Create a match filter on the ingress device. To specify a match filter, include the `filter` *`filter-name`* statement at the `[edit firewall]` hierarchy level. A packet that passes through the filter is compared against a set of rules to classify it and to determine its membership in a set. Once classified, the packet is forwarded to a routing table specified in the accept action in the filter description language. The routing table then forwards the packet to the next hop that corresponds to the destination address entry in the table.

- Create routing instances that specify the routing table(s) to which a packet is forwarded, and the destination to which the packet is forwarded at the `[edit routing-instances]` hierarchy level. For example:

```
[edit]
routing-instances {
    routing-table-name1 {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 172.16.0.14;
            }
        }
    }
    routing-table-name2 {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 172.16.0.18;
            }
        }
    }
}
```

- Create a RIB group to share interface routes with the forwarding routing instances used in filter-based forwarding (FBF). This part of the configuration resolves the routes installed in the routing instances to directly connected next hops on that interface. Create the routing table group at the `[edit routing-options]` hierarchy level.

```
[edit]
routing-options {
    interface-routes {
        rib-group;
        inet {
```

```
            int-routes;
            }
        }
    }
}
routing-options {
    rib-groups {
        int-routes {
        import-rib {
            inet.0;
            webtraffic.inet.0;
            }
        }
    }
}
```

This example shows a packet filter that directs customer traffic to a next-hop router in the domains, SP1 or SP2, based on the packet's source address.

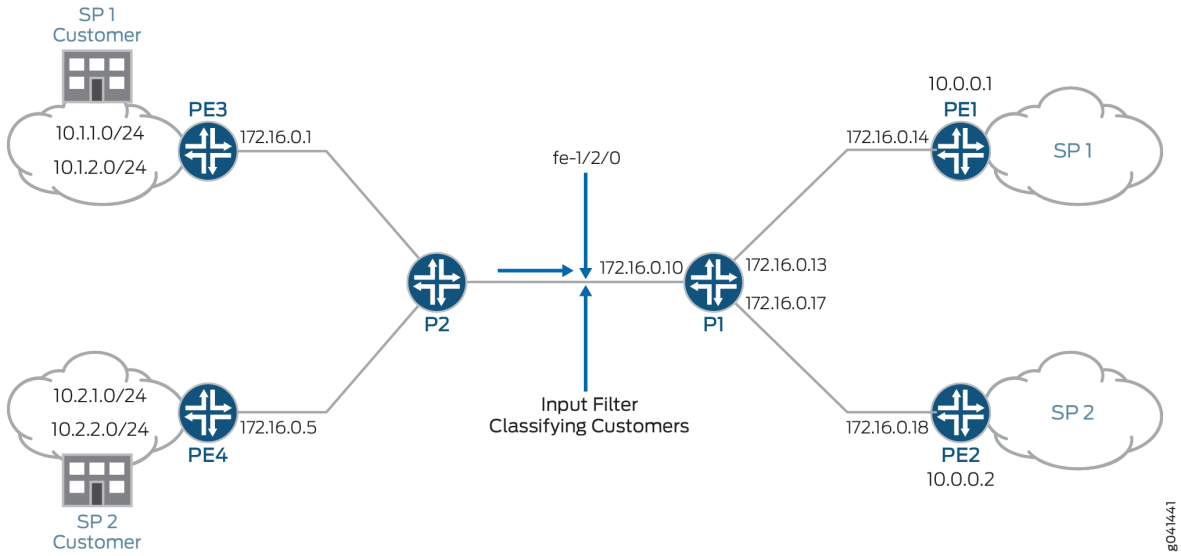
If the packet has a source address assigned to an SP1 customer, destination-based forwarding occurs using the sp1-route-table.inet.0 routing table. If the packet has a source address assigned to an SP2 customer, destination-based forwarding occurs using the sp2-route-table.inet.0 routing table. If a packet does not match either of these conditions, the filter accepts the packet, and destination-based forwarding occurs using the standard inet.0 routing table.

Topology

Figure 66 on page 1501 shows the topology used in this example.

On Device P1, an input filter classifies packets received from Device PE3 and Device PE4. The packets are routed based on the source addresses. Packets with source addresses in the 10.1.1.0/24 and 10.1.2.0/24 networks are routed to Device PE1. Packets with source addresses in the 10.2.1.0/24 and 10.2.2.0/24 networks are routed to Device PE2.

Figure 66: Filter-Based Forwarding



To establish connectivity, OSPF is configured on all of the interfaces. For demonstration purposes, loopback interface addresses are configured on the routing devices to represent networks in the clouds.

The "CLI Quick Configuration" on page 1501 section shows the entire configuration for all of the devices in the topology. The "Configuring Filter-Based Forwarding on Device P1" on page 1505 section shows the step-by-step configuration of the ingress routing device, Device P1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1501**
- Configuring the Firewall Filter on P1 | **1503**
- Configuring Filter-Based Forwarding on Device P1 | **1505**
- Results | **1506**

CLI Quick Configuration

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

Device P1

```
set firewall filter classify-customers term sp1-customers from source-address 10.1.1.0/24
set firewall filter classify-customers term sp1-customers from source-address 10.1.2.0/24
set firewall filter classify-customers term sp1-customers then log
set firewall filter classify-customers term sp1-customers then routing-instance sp1-route-table
set firewall filter classify-customers term sp2-customers from source-address 10.2.1.0/24
set firewall filter classify-customers term sp2-customers from source-address 10.2.2.0/24
set firewall filter classify-customers term sp2-customers then log
set firewall filter classify-customers term sp2-customers then routing-instance sp2-route-table
set firewall filter classify-customers term default then accept
set interfaces fe-1/2/0 unit 0 family inet filter input classify-customers
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.10/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.0.13/30
set interfaces fe-1/2/2 unit 0 family inet address 172.16.0.17/30
set protocols ospf rib-group fbf-group
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances sp1-route-table instance-type forwarding
set routing-instances sp1-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.14
set routing-instances sp2-route-table instance-type forwarding
set routing-instances sp2-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.18
set routing-options interface-routes rib-group fbf-group
set routing-options rib-groups fbf-group import-rib inet.0
set routing-options rib-groups fbf-group import-rib sp1-route-table.inet.0
set routing-options rib-groups fbf-group import-rib sp2-route-table.inet.0
```

Device P2

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.0.6/30
set interfaces fe-1/2/2 unit 0 family inet address 172.16.0.9/30
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE1

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.14/30
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
```

```
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE2

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.18/30
set interfaces lo0 unit 0 family inet address 172.16.2.2/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE3

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.1/30
set interfaces lo0 unit 0 family inet address 10.1.1.1/32
set interfaces lo0 unit 0 family inet address 10.1.2.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE4

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.5/30
set interfaces lo0 unit 0 family inet address 10.2.1.1/32
set interfaces lo0 unit 0 family inet address 10.2.2.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Configuring the Firewall Filter on P1

Step-by-Step Procedure

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

To configure the firewall filter on the main router or switch:

1. Configure the source addresses for SP1 customers.

```
[edit firewall filter classify-customers term sp1-customers]
user@host# set from source-address 10.1.1.0/24
user@host# set from source-address 10.1.2.0/24
```

2. Configure the actions that are taken when packets are received with the specified source addresses; they are logged, and they are passed to the sp1-route-table routing instance for routing via the sp1-route-table.inet.0 routing table.

```
[edit firewall filter classify-customers term sp1-customers]
user@host# set then log
user@host# set then routing-instance sp1-route-table
```

3. Configure the source addresses for SP2 customers.

```
[edit firewall filter classify-customers term sp2-customers]
user@host# set from source-address 10.2.1.0/24
user@host# set from source-address 10.2.2.0/24
```

4. Configure the actions that are taken when packets are received with the specified source addresses; they are logged, and they are passed to the sp2-route-table routing instance for routing via the sp2-route-table.inet.0 routing table.

```
[edit firewall filter classify-customers term sp2-customers]
user@host# set then log
user@host# set then routing-instance sp2-route-table
```

5. Configure the action to take when packets are received from any other source address; they are accepted and routed using the default IPv4 unicast routing table, inet.0.

```
[edit firewall filter classify-customers term default]
user@host# set then accept
```

Configuring Filter-Based Forwarding on Device P1

Step-by-Step Procedure

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

To configure the routing instances:

1. Configure the interfaces.

```
[edit interfaces fe-1/2/0]
user@host# set unit 0 family inet address 172.16.0.10/30
[edit interfaces fe-1/2/1]
user@host# set unit 0 family inet address 172.16.0.13/30
[edit interfaces fe-1/2/2]
user@host# set unit 0 family inet address 172.16.0.17/30
```

2. Assign the `classify-customers` firewall filter to router interface fe-1/2/0.0 as an input packet filter.

```
[edit interfaces fe-1/2/0]
user@host# set unit 0 family inet filter input classify-customers
```

3. Configure connectivity, using either a routing protocol or static routing.

   As a best practice, disable routing on the management interface.

```
[edit  protocols ospf area 0.0.0.0]
user@host# set interface all
user@host# set interface fxp0.0 disable
```

4. Create the routing instances that are referenced in the `classify-customers` firewall filter. The forwarding instance type provides support for filter-based forwarding, where interfaces are not associated with instances.

```
[edit routing-instances]
user@host# set sp1-route-table instance-type forwarding
user@host# set sp2-route-table instance-type forwarding
```

5. For each routing instance, define a default route to forward traffic to the specified next hop (PE1 and PE2 in this example).

```
[edit routing-instances ]
user@host# set sp1-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.14
user@host# set sp2-route-table routing-options static route 0.0.0.0/0 next-hop 172.16.0.18
```

6. Associate the routing tables to form a routing table group. The first routing table, inet.0, is the primary routing table, and the others are secondary routing tables. The primary routing table determines the address family of the routing table group, in this case IPv4.

```
[edit routing-options]
user@host# set rib-groups fbf-group import-rib inet.0
user@host# set rib-groups fbf-group import-rib sp1-route-table.inet.0
user@host# set rib-groups fbf-group import-rib sp2-route-table.inet.0
```

7. Specify the fbf-group routing table group within the OSPF configuration to install OSPF routes into the three routing tables.

```
[edit protocols ospf]
user@host# set rib-group fbf-group
```

8. Commit the configuration when you are done.

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

Results

Confirm your configuration by issuing the `show interfaces`, `show firewall`, `show protocols`, `show routing-instances`, and `show routing-options` commands.

```
user@host# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            filter {
                input classify-customers;
```

```
            }
            address 172.16.0.10/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 172.16.0.13/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        family inet {
            address 172.16.0.17/30;
        }
    }
}
```

```
user@host# show firewall
filter classify-customers {
    term sp1-customers {
        from {
            source-address {
                10.1.1.0/24;
                10.1.2.0/24;
            }
        }
        then {
            log;
            routing-instance sp1-route-table;
        }
    }
    term sp2-customers {
        from {
            source-address {
                10.2.1.0/24;
                10.2.2.0/24;
            }
        }
```

```
        then {
            log;
            routing-instance sp2-route-table;
        }
    }
    term default {
        then accept;
    }
}
```

```
user@host# show protocols
ospf {
    rib-group fbf-group;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

```
user@host# show routing-instances
sp1-route-table {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.16.0.14;
        }
    }
}
sp2-route-table {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.16.0.18;
        }
```

```
    }
}
```

```
user@host# show routing-options
rib-groups {
    fbf-group {
        import-rib [ inet.0 sp1-route-table.inet.0 sp2-route-table.inet.0 ];
    }
}
```

Verification

IN THIS SECTION

- Pinging with Specified Source Addresses | **1509**
- Verifying the Firewall Filter | **1510**

Confirm that the configuration is working properly.

Pinging with Specified Source Addresses

Purpose

Send some ICMP packets across the network to test the firewall filter.

Action

1. Run the `ping` command, pinging the lo0.0 interface on Device PE1.

   The address configured on this interface is 172.16.1.1.

   Specify the source address 10.1.2.1, which is the address configured on the lo0.0 interface on Device PE3.

```
user@PE3> ping 172.16.1.1 source 10.1.2.1
PING 172.16.1.1 (172.16.1.1): 56 data bytes
64 bytes from 172.16.1.1: icmp_seq=0 ttl=62 time=1.444 ms
```

```
64 bytes from 172.16.1.1: icmp_seq=1 ttl=62 time=2.094 ms
^C
--- 172.16.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.444/1.769/2.094/0.325 ms
```

2. Run the `ping` command, pinging the lo0.0 interface on Device PE2.

   The address configured on this interface is 172.16.2.2.

   Specify the source address 10.2.1.1, which is the address configured on the lo0.0 interface on Device PE4.

```
user@PE4> ping 172.16.2.2 source 10.2.1.1
PING 172.16.2.2 (172.16.2.2): 56 data bytes
64 bytes from 172.16.2.2: icmp_seq=0 ttl=62 time=1.473 ms
64 bytes from 172.16.2.2: icmp_seq=1 ttl=62 time=1.407 ms
^C
--- 172.16.2.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.407/1.440/1.473/0.033 ms
```

Meaning

Sending these pings activates the firewall filter actions.

Verifying the Firewall Filter

Purpose

Make sure the firewall filter actions take effect.

Action

1. Run the `show firewall log` command on Device P1.

```
user@P1> show firewall log
Log :
Time      Filter    Action Interface     Protocol        Src Addr           Dest Addr
13:52:20  pfe       A      fe-1/2/0.0    ICMP            10.2.1.1           172.16.2.2
```

```
13:52:19  pfe       A       fe-1/2/0.0   ICMP          10.2.1.1          172.16.2.2
13:51:53  pfe       A       fe-1/2/0.0   ICMP          10.1.2.1          172.16.1.1
13:51:52  pfe       A       fe-1/2/0.0   ICMP          10.1.2.1          172.16.1.1
```

RELATED DOCUMENTATION

- Configuring Filter-Based Forwarding
- Copying and Redirecting Traffic with Port Mirroring and Filter-Based Forwarding
- Using Filter-Based Forwarding to Export Monitored Traffic to Multiple Destinations
- Filter-Based Forwarding Overview | **1493**

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address

IN THIS SECTION

- Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**
- Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface | **1513**
- Example: Configuring Filter-Based Forwarding to a Specific Destination IP Address | **1520**

Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address

Policy-based routing (also known as filter-based forwarding) refers to the use of firewall filters that are applied to an interface to match certain IP header characteristics and to route only those matching packets differently than the packets would normally be routed.

Starting in Junos OS Release 12.2, you can use `then next-interface`, `then next-ip`, or `then next-ip6` as an action in a *firewall filter*. From specific match conditions, IPv4 and IPv6 addresses or an interface name can be specified as the response action to a match.

The set of match conditions can be as follows:

- Layer-3 properties (for example, the source or destination IP address or the TOS byte)

- Layer-4 properties (for example, the source or destination port)

The route for the given IPv4 or IPv6 address has to be present in the routing table for policy-based routing to take effect. Similarly, the route through the given interface has to be present in the forwarding table for `next-interface` action to take effect. This can be achieved by configuring an interior gateway protocol (IGP), such as OSPF or IS-IS, to advertise Layer 3 routes.

The firewall filter matches the conditions and forwards the packet to one of the following:

- An IPv4 address (using the `next-ip` firewall filter action)
- An IPv6 address (using the `next-ip6` firewall filter action)
- An interface (using the `next-interface` firewall filter action)

Suppose, for example, that you want to offer services to your customers, and the services reside on different servers. An example of a service might be hosted DNS or hosted FTP. As customer traffic arrives at the Juniper Networks routing device, you can use filter-based forwarding to send traffic to the servers by applying a match condition on a MAC address or an IP address or simply an incoming interface and send the packets to a certain outgoing interface that is associated with the appropriate server. Some of your destinations might be IPv4 or IPv6 addresses, in which case the `next-ip` or `next-ip6` action is useful.

Optionally, you can associate the outgoing interfaces or IP addresses with routing instances.

For example:

```
firewall {
    filter filter1 {
        term t1 {
            from {
                source-address {
                    10.1.1.3/32;
                }
            }
            then {

                next-interface {
                    xe-0/1/0.1;
                    routing-instance rins1;
                }
            }
        }
        term t2 {
            from {
```

```
                source-address {
                    10.1.1.4/32;
                }
            }
            then {
                next-interface {
                    xe-0/1/0.2;
                    routing-instance rins2;
                }
            }
        }
    }
}
routing-instances {
    rins1 {
        instance-type virtual-router;
        interface xe-0/1/0.1;
    }
    rins2 {
        instance-type virtual-router;
        interface xe-0/1/0.2;
    }
}
```

SEE ALSO

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address | **1511**

Firewall Filter Nonterminating Actions | **871**

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface

IN THIS SECTION

- Requirements | **1514**
- Overview | **1514**
- Configuration | **1515**
- Verification | **1519**

This example shows how to use `then next-interface` as an action in a firewall filter.

Requirements

This example has the following hardware and software requirements:

- MX Series 5G Universal Routing Platform as the routing device with the firewall filter configured.
- Junos OS Release 12.2 running on the routing device with the firewall filter configured.
- The filter with the `next-interface` (or `next-ip`) action can only be applied to an interface that is hosted on a Trio MPC. If you apply the filter to an I-chip based DPC, the commit operation fails.
- The outgoing interface referred to in the `next-interface` *`interface-name`* action can be hosted on a Trio MPC or an I-chip based DPC.

Overview

IN THIS SECTION

- Topology | **1515**

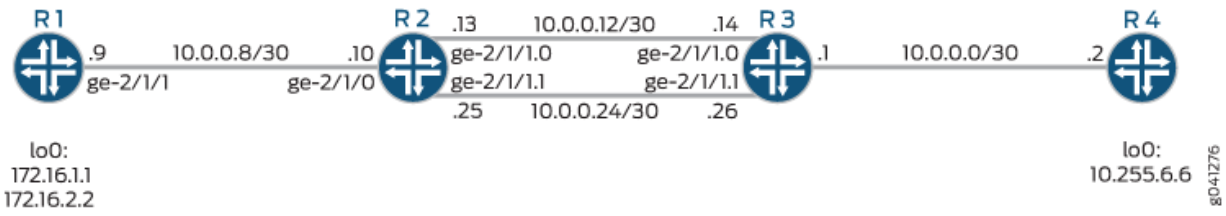
In this example, Device R1 has two loopback interface addresses configured: 172.16.1.1 and 172.16.2.2.

On Device R2, a firewall filter has multiple terms configured. Each term matches one of the source addresses in incoming traffic, and routes the traffic to specified outgoing interfaces. The outgoing interfaces are configured as VLAN-tagged interfaces between Device R2 and Device R3.

IS-IS is used for connectivity among the devices.

Figure 67 on page 1514 shows the topology used in this example.

Figure 67: Filter-Based Forwarding to Specified Outgoing Interfaces



This example shows the configuration on Device R2.

Topology

Configuration

IN THIS SECTION

- Procedure | 1515

Procedure

CLI Quick Configuration

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

Device R2

```
set interfaces ge-2/1/0 unit 0 family inet filter input filter1
set interfaces ge-2/1/0 unit 0 family inet address 10.0.0.10/30
set interfaces ge-2/1/0 unit 0 description to-R1
set interfaces ge-2/1/0 unit 0 family iso
set interfaces ge-2/1/1 vlan-tagging
set interfaces ge-2/1/1 description to-R3
set interfaces ge-2/1/1 unit 0 vlan-id 1001
set interfaces ge-2/1/1 unit 0 family inet address 10.0.0.13/30
set interfaces ge-2/1/1 unit 0 family iso
set interfaces ge-2/1/1 unit 1 vlan-id 1002
set interfaces ge-2/1/1 unit 1 family inet address 10.0.0.25/30
set interfaces ge-2/1/1 unit 1 family iso
set interfaces lo0 unit 0 family inet address 10.255.4.4/32
set interfaces lo0 unit 0 family iso address 49.0001.0010.0000.0404.00
set firewall family inet filter filter1 term t1 from source-address 172.16.1.1/32
set firewall family inet filter filter1 term t1 then next-interface ge-2/1/1.0
set firewall family inet filter filter1 term t2 from source-address 172.16.2.2/32
set firewall family inet filter filter1 term t2 then next-interface ge-2/1/1.1
set protocols isis interface all level 1 disable
```

```
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set ge-2/1/0 unit 0 family inet filter input filter1
user@R2# set ge-2/1/0 unit 0 family inet address 10.0.0.10/30
user@R2# set ge-2/1/0 unit 0 description to-R1
user@R2# set ge-2/1/0 unit 0 family iso
user@R2# set ge-2/1/1 vlan-tagging
user@R2# set ge-2/1/1 description to-R3
user@R2# set ge-2/1/1 unit 0 vlan-id 1001
user@R2# set ge-2/1/1 unit 0 family inet address 10.0.0.13/30
user@R2# set ge-2/1/1 unit 0 family iso
user@R2# set ge-2/1/1 unit 1 vlan-id 1002
user@R2# set ge-2/1/1 unit 1 family inet address 10.0.0.25/30
user@R2# set ge-2/1/1 unit 1 family iso
user@R2# set lo0 unit 0 family inet address 10.255.4.4/32
user@R2# set lo0 unit 0 family iso address 49.0001.0010.0000.0404.00
```

2. Configure the firewall filter.

```
[edit firewall family inet filter filter1]
user@R2# set term t1 from source-address 172.16.1.1/32
user@R2# set term t1 then next-interface ge-2/1/1.0
user@R2# set term t2 from source-address 172.16.2.2/32
user@R2# set term t2 then next-interface ge-2/1/1.1
```

3. Enable IS-IS on the interfaces.

```
[edit protocols is-is]
user@R2# set interface all level 1 disable
user@R2# set interface fxp0.0 disable
user@R2# set interface lo0.0
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show firewall`, and `show protocols` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@R2# show interfaces
ge-2/1/0 {
    unit 0 {
        description to-R1;
        family inet {
            filter {
                input filter1;
            }
            address 10.0.0.10/30;
        }
        family iso;
    }
}
ge-2/1/1 {
    description to-R3;
    vlan-tagging;
    unit 0 {
        vlan-id 1001;
        family inet {
            address 10.0.0.13/30;
        }
        family iso;
    }
    unit 1 {
        vlan-id 1002;
        family inet {
            address 10.0.0.25/30;
```

```
        }
        family iso;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.4.4/32;
        }
        family iso {
            address 49.0001.0010.0000.0404.00;
        }
    }
}
```

```
user@R2# show firewall
family inet {
    filter filter1 {
        term t1 {
            from {
                source-address {
                    172.16.1.1/32;
                }
            }
            then {
                next-interface {
                ge-2/1/1.0;
            }
        }
        term t2 {
            from {
                source-address {
                    172.16.2.2/32;
                }
            }
            then {
                next-interface {
                    ge-2/1/1.1;
                }
            }
        }
```

```
    }
}
```

```
user@R2# show protocols
isis {
    interface all {
        level 1 disable;
    }
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Checking the Paths Used | **1519**

Confirm that the configuration is working properly.

Checking the Paths Used

Purpose

Make sure that the expected paths are used when sending traffic from Device R1 to Device R4.

Action

On Device R1, enter the `traceroute` command.

```
user@R1> traceroute 10.255.6.6 source 172.16.1.1
traceroute to 10.255.6.6 (10.255.6.6) from 172.16.1.1, 30 hops max, 40 byte packets
 1  10.0.0.10 (10.0.0.10)  0.976 ms  0.895 ms  0.815 ms
```

```
 2  10.0.0.14 (10.0.0.14)  0.868 ms  0.888 ms  0.813 ms
 3  10.255.6.6 (10.255.6.6)  1.715 ms  1.442 ms  1.382 ms
```

```
user@R1> traceroute 10.255.6.6 source 172.16.2.2
traceroute to 10.255.6.6 (10.255.6.6) from 172.16.2.2, 30 hops max, 40 byte packets
 1  10.0.0.10 (10.0.0.10)  0.973 ms  0.907 ms  0.782 ms
 2  10.0.0.26 (10.0.0.26)  0.844 ms  0.890 ms  0.852 ms
 3  10.255.6.6 (10.255.6.6)  1.384 ms  1.516 ms  1.462 ms
```

Meaning

The output shows that the second hop changes, depending on the source address used in the `traceroute` command.

To verify this feature, a traceroute operation is performed on Device R1 to Device R4. When the source IP address is 172.16.1.1, packets are forwarded out the ge-2/1/1.0 interface on Device R2. When the source IP address is 172.16.2.2, packets are forwarded out the ge-2/1/1.1 interface on Device R2.

SEE ALSO

Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address

Example: Configuring Filter-Based Forwarding to a Specific Destination IP Address

Firewall Filter Nonterminating Actions | 871

Example: Configuring Filter-Based Forwarding to a Specific Destination IP Address

IN THIS SECTION

- Requirements | 1521
- Overview | 1521
- Configuration | 1522
- Verification | 1531

This example shows how to use `then next-ip` as an action in a firewall filter.

Requirements

This example has the following hardware and software requirements:

- MX Series 5G Universal Routing Platform as the routing device with the firewall filter configured.
- Junos OS Release 12.2 running on the routing device with the firewall filter configured.
- The filter with the `next-interface` (or `next-ip`) action can only be applied to an interface that is hosted on a Trio MPC. If you apply the filter to an I-chip based DPC, the commit operation fails.
- The outgoing interface referred to in the next-interface interface-name action can be hosted on a Trio MPC or an I-chip based DPC.

Overview

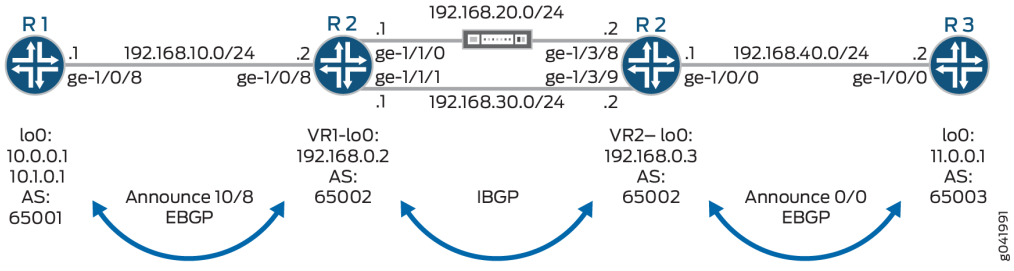
IN THIS SECTION

- Topology | **1522**

In this example, Device R2 has two routing instances that are interconnected with physical links. Traffic from certain sources is required to be directed across the upper link for inspection by a traffic optimizer, which acts transparently on the IP layer. When the traffic optimizer fails, the traffic moves to the lower link. Flows in direction R1>R3 and R3>R1 follow identical paths.

Figure 68 on page 1521 shows the topology used in this example.

Figure 68: Filter-Based Forwarding to Specified Outgoing Interfaces



On Device R2, a firewall filter is applied to interface ge-1/0/8 in the input direction. The second term matches the specific source addresses 10.0.0.0/24, and routes the traffic to address 192.168.0.3. This

address resolves to next-hop 192.168.20.2. If the link connected to interface ge-1/1/0 goes down, the address 192.168.0.3 will resolve to next-hop 192.168.30.2.

On Device R2, a firewall filter is applied to interface ge-1/0/0 in the input direction. The second term matches the specific destination addresses 10.0.0.0/24, and routes the traffic to address 192.168.0.2. This address resolves to next-hop 192.168.20.1. If the link connected to interface ge-1/3/8 goes down, the address 192.168.0.2 will resolve to next-hop 192.168.30.1.

NOTE: The address configured using the `next-ip` action is not automatically resolved. On Ethernet interfaces, it is assumed that the configured address is resolved using a routing protocol or static routes.

Internal BGP (IBGP) is used between Device R2-VR1 and Device R2-VR2. External BGP (EBGP) is used between Device R1 and Device R2-VR1, as well as between Device R2-VR2 and Device R3.

BGP operations proceed as follows:

- R2-VR1 learns 10/8 from R1, and 0/0 from R2-VR2.
- R2-VR2 learns 0/0 from R3, and 10/8 from R2-VR1.
- R1 advertises 10/8, and receives 0/0 from R2-VR1.
- R3 advertises 0/0, and receives 10/8 from R2-VR2.

The firewall filter applied to Device R2 needs to allow control-plane traffic for the directly connected interfaces, in this case the EBGP sessions.

This example shows the configuration on Device R2.

Topology

Configuration

IN THIS SECTION

- Procedure | **1523**

Procedure

CLI Quick Configuration

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

Device R1

```
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces lo0 unit 0 family inet address 10.1.0.1/32
set interfaces ge-1/0/8 unit 0 family inet address 192.168.10.1/24
set routing-options autonomous-system 64501
set protocols bgp group eBGP neighbor 192.168.10.2 peer-as 64502
set protocols bgp group eBGP export Announce10
set policy-options policy-statement Announce10 term 1 from route-filter 10.0.0.0/8 exact
set policy-options policy-statement Announce10 term 1 then accept
set policy-options policy-statement Announce10 term 2 then reject
```

Device R2

```
set interfaces ge-1/0/8 unit 0 family inet address 192.168.10.2/24
set interfaces ge-1/0/8 unit 0 family inet filter input SteerSrcTrafficOptimizer
set interfaces ge-1/1/0 unit 0 family inet address 192.168.20.1/24
set interfaces ge-1/1/1 unit 0 family inet address 192.168.30.1/24
set routing-instances VR1 instance-type virtual-router
set routing-instances VR1 interface ge-1/0/8.0
set routing-instances VR1 interface ge-1/1/0.0
set routing-instances VR1 interface ge-1/1/1.0
set routing-instances VR1 routing-options static route 192.168.0.3 next-hop 192.168.20.2
set routing-instances VR1 routing-options static route 192.168.0.3 qualified-next-hop
192.168.30.2 metric 100
set routing-instances VR1 routing-options autonomous-system 64502
set routing-instances VR1 protocols bgp group eBGP neighbor 192.168.10.1 peer-as 64501
set routing-instances VR1 protocols bgp group iBGP neighbor 192.168.30.2 peer-as 64502
set routing-instances VR1 protocols bgp group iBGP neighbor 192.168.30.2 export AcceptExternal
set firewall family inet filter SteerSrcTrafficOptimizer term 0 from source-address
192.168.10.0/24
set firewall family inet filter SteerSrcTrafficOptimizer term 0 then accept
set firewall family inet filter SteerSrcTrafficOptimizer term 1 from source-address 10.0.0.0/24
set firewall family inet filter SteerSrcTrafficOptimizer term 1 then next-ip 192.168.0.3 routing-
```

```
instance VR1
set firewall family inet filter SteerSrcTrafficOptimizer term 2 from source-address 10.0.0.0/8
set firewall family inet filter SteerSrcTrafficOptimizer term 2 then accept
set interfaces ge-1/0/0 unit 0 family inet address 192.168.40.1/24
set interfaces ge-1/0/0 unit 0 family inet filter input SteerDstTrafficOptimizer
set interfaces ge-1/3/8 unit 0 family inet address 192.168.20.2/24
set interfaces ge-1/3/9 unit 0 family inet address 192.168.30.2/24
set routing-instances VR2 instance-type virtual-router
set routing-instances VR2 interface ge-1/0/0.0
set routing-instances VR2 interface ge-1/3/8.0
set routing-instances VR2 interface ge-1/3/9.0
set routing-instances VR2 routing-options static route 192.168.0.2/32 next-hop 192.168.20.1
set routing-instances VR2 routing-options static route 192.168.0.2/32 qualified-next-hop
192.168.30.1 metric 100
set routing-instances VR2 routing-options autonomous-system 64502
set routing-instances VR2 protocols bgp group eBGP neighbor 192.168.40.2 peer-as 64503
set routing-instances VR2 protocols bgp group iBGP neighbor 192.168.30.1 peer-as 64502
set routing-instances VR2 protocols bgp group iBGP neighbor 192.168.30.1 export AcceptExternal
set firewall family inet filter SteerDstTrafficOptimizer term 0 from source-address
192.168.40.0/24
set firewall family inet filter SteerDstTrafficOptimizer term 0 then accept
set firewall family inet filter SteerDstTrafficOptimizer term 1 from destination-address
10.0.0.0/24
set firewall family inet filter SteerDstTrafficOptimizer term 1 then next-ip 192.168.0.2 routing-
instance VR2
set firewall family inet filter SteerDstTrafficOptimizer term 2 from destination-address
10.0.0.0/8
set firewall family inet filter SteerDstTrafficOptimizer term 2 then accept
set policy-options policy-statement AcceptExternal term 1 from route-type external
set policy-options policy-statement AcceptExternal term 1 then accept
```

Device R3

```
set interfaces lo0 unit 0 family inet address 10.11.0.1/32
set interfaces ge-1/0/0 unit 0 family inet address 192.168.40.2/24
set routing-options autonomous-system 64503
set protocols bgp group eBGP neighbor 192.168.40.1 peer-as 64502
set protocols bgp group eBGP export Announce0
set policy-options policy-statement Announce0 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement Announce0 term 1 then accept
set policy-options policy-statement Announce0 term 2 then reject
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889 in the Junos OS CLI User Guide.

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set ge-1/0/8 unit 0 family inet address 192.168.10.2/24
user@R2# set ge-1/0/8 unit 0 family inet filter input SteerSrcTrafficOptimizer
user@R2# set ge-1/1/0 unit 0 family inet address 192.168.20.1/24
user@R2# set ge-1/1/1 unit 0 family inet address 192.168.30.1/24
user@R2# set ge-1/0/0 unit 0 family inet address 192.168.40.1/24
user@R2# set ge-1/0/0 unit 0 family inet filter input SteerDstTrafficOptimizer
user@R2# set ge-1/3/8 unit 0 family inet address 192.168.20.2/24
user@R2# set ge-1/3/9 unit 0 family inet address 192.168.30.2/24
```

2. Configure the routing instance.

```
[edit routing-instances]
user@R2# set VR1 instance-type virtual-router
user@R2# set VR1 interface ge-1/0/8.0
user@R2# set VR1 interface ge-1/1/0.0
user@R2# set VR1 interface ge-1/1/1.0
user@R2# set VR2 instance-type virtual-router
user@R2# set VR2 interface ge-1/0/0.0
user@R2# set VR2 interface ge-1/3/8.0
user@R2# set VR2 interface ge-1/3/9.0
```

3. Configure the static and BGP routing.

```
[edit routing-instances]
user@R2# set VR1 routing-options static route 192.168.0.3 next-hop 192.168.20.2
user@R2# set VR1 routing-options static route 192.168.0.3 qualified-next-hop 192.168.30.2
metric 100
user@R2# set VR1 routing-options autonomous-system 64502
user@R2# set VR1 protocols bgp group eBGP neighbor 192.168.10.1 peer-as 64501
user@R2# set VR1 protocols bgp group iBGP neighbor 192.168.30.2 peer-as 64502
```

```
user@R2# set VR1 protocols bgp group iBGP neighbor 192.168.30.2 export AcceptExternal
user@R2# set VR2 routing-options static route 192.168.0.2/32 next-hop 192.168.20.1
user@R2# set VR2 routing-options static route 192.168.0.2/32 qualified-next-hop 192.168.30.1
metric 100
user@R2# set VR2 routing-options autonomous-system 64502
user@R2# set VR2 protocols bgp group eBGP neighbor 192.168.40.2 peer-as 64503
user@R2# set VR2 protocols bgp group iBGP neighbor 192.168.30.1 peer-as 64502
user@R2# set VR2 protocols bgp group iBGP neighbor 192.168.30.1 export AcceptExternal
```

4. Configure the firewall filters.

```
[edit firewall family inet]
user@R2# set filter SteerSrcTrafficOptimizer term 0 from source-address 192.168.10.0/24
user@R2# set filter SteerSrcTrafficOptimizer term 0 then accept
user@R2# set filter SteerSrcTrafficOptimizer term 1 from source-address 10.0.0.0/24
user@R2# set filter SteerSrcTrafficOptimizer term 1 then next-ip 192.168.0.3 routing-instance
VR1
user@R2# set filter SteerSrcTrafficOptimizer term 2 from source-address 10.0.0.0/8
user@R2# set filter SteerSrcTrafficOptimizer term 2 then accept
user@R2# set filter SteerDstTrafficOptimizer term 0 from source-address 192.168.40.0/24
user@R2# set filter SteerDstTrafficOptimizer term 0 then accept
user@R2# set filter SteerDstTrafficOptimizer term 1 from destination-address 10.0.0.0/24
user@R2# set filter SteerDstTrafficOptimizer term 1 then next-ip 192.168.0.2 routing-instance
VR2
user@R2# set filter SteerDstTrafficOptimizer term 2 from destination-address 10.0.0.0/8
user@R2# set filter SteerDstTrafficOptimizer term 2 then accept
```

5. Configure the routing policy.

```
[edit policy-options policy-statement AcceptExternal term 1]
user@R2# set from route-type external
user@R2# set term 1 then accept
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show firewall`, and `show protocols` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@R2# show interfaces
ge-1/0/0 {
    unit 0 {
        family inet {
            filter {
                input SteerDstTrafficOptimizer;
            }
            address 192.168.40.1/24;
        }
    }
}
ge-1/0/8 {
    unit 0 {
        family inet {
            filter {
                input SteerSrcTrafficOptimizer;
            }
            address 192.168.10.2/24;
        }
    }
}
ge-1/1/0 {
    unit 0 {
        family inet {
            address 192.168.20.1/24;
        }
    }
}
ge-1/1/1 {
    unit 0 {
        family inet {
            address 192.168.30.1/24;
        }
    }
}
ge-1/3/8 {
```

```
    unit 0 {
        family inet {
            address 192.168.20.2/24;
        }
    }
}
ge-1/3/9 {
    unit 0 {
        family inet {
            address 192.168.30.2/24;
        }
    }
}
```

```
user@R2# show firewall
family inet {
    filter SteerSrcTrafficOptimizer {
        term 0 {
            from {
                source-address {
                    192.168.10.0/24;
                }
            }
            then accept;
        }
        term 1 {
            from {
                source-address {
                    10.0.0.0/24;
                }
            }
            then {
                next-ip 192.168.0.3/32 routing-instance VR1;
            }
        }
        term 2 {
            from {
                source-address {
                    10.0.0.0/8;
                }
            }
```

```
            then accept;
        }
    }
    filter SteerDstTrafficOptimizer {
        term 0 {
            from {
                source-address {
                    192.168.40.0/24;
                }
            }
            then accept;
        }
        term 1 {
            from {
                destination-address {
                    10.0.0.0/24;
                }
            }
            then {
                next-ip 192.168.0.2/32 routing-instance VR2;
            }
        }
        term 2 {
            from {
                destination-address {
                    10.0.0.0/8;
                }
            }
            then accept;
        }
    }
}
```

```
user@R2# show policy-options
policy-statement AcceptExternal {
    term 1 {
        from route-type external;
        then accept;
```

```
    }
}
```

```
user@R2# show routing-instances
VR1 {
    instance-type virtual-router;
    interface ge-1/0/8.0;
    interface ge-1/1/0.0;
    interface ge-1/1/1.0;
    routing-options {
        static {
            route 192.168.0.3/32 {
                next-hop 192.168.20.2;
                qualified-next-hop 192.168.30.2 {
                    metric 100;
                }
            }
        }
        autonomous-system 64502;
    }
    protocols {
        bgp {
            group eBGP {
                neighbor 192.168.10.1 {
                    peer-as 64501;
                }
            }
            group iBGP {
                neighbor 192.168.30.2 {
                    export NextHopSelf;
                    peer-as 64502;
                }
            }
        }
    }
}
VR2 {
    instance-type virtual-router;
    interface ge-1/0/0.0;
    interface ge-1/3/8.0;
    interface ge-1/3/9.0;
```

```
    routing-options {
        static {
            route 192.168.0.2/32 {
                next-hop 192.168.20.1;
                qualified-next-hop 192.168.30.1 {
                    metric 100;
                }
            }
        }
        autonomous-system 64502;
    }
    protocols {
        bgp {
            group eBGP {
                neighbor 192.168.40.2 {
                    peer-as 64503;
                }
            }
            group iBGP {
                neighbor 192.168.30.1 {
                    export NextHopSelf;
                    peer-as 64502;
                }
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Checking the Paths Used | **1532**

Confirm that the configuration is working properly.

Checking the Paths Used

Purpose

Make sure that the expected paths are used when sending traffic from Device R1 to Device R3.

Action

On Device R1, enter the `traceroute` command before and after the link failure

Before Failure of the Traffic Optimizer

```
user@R1> traceroute 10.11.0.1 source 10.0.0.1
traceroute to 10.11.0.1 (10.11.0.1) from 10.0.0.1, 30 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  0.519 ms  0.403 ms  0.380 ms
 2  192.168.20.2 (192.168.20.2)  0.404 ms  0.933 ms  0.402 m0
 3  10.11.0.1 (10.11.0.1)  0.709 ms  0.656 ms  0.644 ms
```

```
user@R1> traceroute 10.11.0.1 source 10.1.0.1
traceroute to 10.11.0.1 (10.11.0.1) from 10.1.0.1, 30 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  0.524 ms  0.396 ms  0.380 ms
 2  192.168.30.2 (192.168.30.2)  0.412 ms  0.410 ms  0.911 ms
 3  10.11.0.1 (10.11.0.1)  0.721 ms  0.639 ms  0.659 ms
```

After Failure of the Traffic Optimizer

```
user@R1> traceroute 10.11.0.1 source 10.0.0.1
traceroute to 10.11.0.1 (10.11.0.1) from 10.0.0.1, 30 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  0.506 ms  0.400 ms  0.378 ms
 2  192.168.30.2 (192.168.30.2)  0.433 ms  0.550 ms  0.415 ms
 3  10.11.0.1 (10.11.0.1)  0.723 ms  0.638 ms  0.638 ms
```

```
user@R1> traceroute 10.11.0.1 source 10.1.0.1
traceroute to 10.11.0.1 (10.11.0.1) from 10.1.0.1, 30 hops max, 40 byte packets
 1  192.168.10.2 (192.168.10.2)  0.539 ms  0.411 ms  0.769 ms
 2  192.168.30.2 (192.168.30.2)  0.426 ms  0.413 ms  2.429 ms
```

```
3  10.11.0.1 (10.11.0.1)  10.868 ms  0.662 ms  0.647 ms
```

Meaning

The output shows that the second hop changes, depending on the source address used in the `traceroute` command.

To verify this feature, a traceroute operation is performed on Device R1 to Device R3. When the source IP address is 10.0.0.1, packets are forwarded out the ge-1/1/0.0 interface on Device R2. When the source IP address is 10.1.0.1, packets are forwarded out the ge-1/1/1.0 interface on Device R2.

When the link between ge-1/1/0 and ge-1/3/8 fails, packets with source IP address 10.0.0.1 are forwarded out the ge-1/1/1.0 interface on Device R2.

SEE ALSO

- Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address
- Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface
- Firewall Filter Nonterminating Actions | **871**

RELATED DOCUMENTATION

- Example: Configuring Filter-Based Forwarding on Logical Systems | **1236**
- Example: Configuring Filter-Based Forwarding on the Source Address | **1497**
- Firewall Filter Nonterminating Actions | **871**

CHAPTER 26

Configuring Firewall Filters (EX Series Switches)

IN THIS CHAPTER

- Firewall Filters for EX Series Switches Overview | **1535**
- Understanding Planning of Firewall Filters | **1539**
- Understanding Firewall Filter Match Conditions | **1543**
- Understanding How Firewall Filters Control Packet Flows | **1549**
- Understanding How Firewall Filters Are Evaluated | **1550**
- Understanding Firewall Filter Processing Points for Bridged and Routed Packets on EX Series Switches | **1552**
- Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**
- Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches | **1570**
- Support for Match Conditions and Actions for Loopback Firewall Filters on Switches | **1633**
- Configuring Firewall Filters (CLI Procedure) | **1638**
- Understanding How Firewall Filters Test a Packet's Protocol | **1649**
- Understanding Filter-Based Forwarding for EX Series Switches | **1650**
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**
- Example: Configuring a Firewall Filter on a Management Interface on an EX Series Switch | **1680**
- Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**
- Example: Applying Firewall Filters to Multiple Supplicants on Interfaces Enabled for 802.1X or MAC RADIUS Authentication | **1692**
- Verifying That Policers Are Operational | **1699**
- Troubleshooting Firewall Filters | **1700**

Firewall Filters for EX Series Switches Overview

IN THIS SECTION

- Firewall Filter Types | 1535
- Firewall Filter Components | 1536
- Firewall Filter Processing | 1539

Firewall filters provide rules that define whether to permit, deny, or forward packets that are transiting an interface on a Juniper Networks EX Series Ethernet Switch from a source address to a destination address. You configure firewall filters to determine whether to permit, deny, or forward traffic before it enters or exits a port, VLAN, or Layer 3 (routed) interface to which the *firewall filter* is applied. To apply a firewall filter, you must first configure the filter and then apply it to an port, VLAN, or Layer 3 interface.

You can apply firewall filters to network interfaces, aggregated Ethernet interfaces (also known as link aggregation groups (LAGs)), loopback interfaces, management interfaces, virtual management Ethernet interfaces (VMEs), and routed VLAN interfaces (RVIs). For information on EX Series switches that support a firewall filter on these interfaces, see EX Series Switch Software Features Overview.

An *ingress* firewall filter is a filter that is applied to packets that are entering a network. An *egress* firewall filter is a filter that is applied to packets that are exiting a network. You can configure firewall filters to subject packets to filtering, class-of-service (CoS) marking (grouping similar types of traffic together, and treating each type of traffic as a class with its own level of service priority), and traffic policing (controlling the maximum rate of traffic sent or received on an interface).

NOTE: Policers on network port, layer 2 and layer 3, or IRB interfaces do not police host-bound traffic. But if you want to prevent DDoS attacks, then you can create a firewall filter on the lo0 that protects the routing engine.

Firewall Filter Types

The following firewall filter types are supported for EX Series switches:

- Port (Layer 2) firewall filter—Port firewall filters apply to Layer 2 switch ports. You can apply port firewall filters in both ingress and egress directions on a physical port.

- VLAN firewall filter—VLAN firewall filters provide access control for packets that enter a VLAN, are bridged within a VLAN, or leave a VLAN. You can apply VLAN firewall filters in both ingress and egress directions on a VLAN. VLAN firewall filters are applied to all packets that are forwarded to or forwarded from the VLAN.
- Router (Layer 3) firewall filter—You can apply a router firewall filter in both ingress and egress directions on Layer 3 (routed) interfaces and routed VLAN interfaces (RVIs). You can apply a router firewall filter in the ingress direction on the loopback interface (**lo0**) also. Firewall filters configured on loopback interfaces are applied only to packets that are sent to the Routing Engine CPU for further processing.

You can apply port, VLAN, or router firewall filters to *both* IPv4 and IPv6 traffic on these switches:

- EX2200 switch
- EX3300 switch
- EX3200 switch
- EX4200 switch
- EX4300 switch
- EX4400 switch
- EX4500 switch
- EX4550 switch
- EX6200 switch
- EX8200 switch

For information on firewall filters supported on different switches, see "Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches" on page 1570.

Firewall Filter Components

In a firewall filter, you first define the family address type (**ethernet-switching**, **inet**, or **inet6**), and then you define one or more terms that specify the filtering criteria (specified as terms with match conditions) and the action (specified as actions or action modifiers) to take if a match occurs.

The maximum number of terms allowed per firewall filter for EX Series switches is:

- 512 for EX2200 switches
- 1436 for EX3300 switches

NOTE: On EX3300 switches, if you add and delete filters with a large number of terms (on the order of 1000 or more) in the same commit operation, not all the filters are installed. You must add filters in one commit operation, and delete filters in a separate commit operation.

- 7,042 for EX3200 and EX4200 switches—as allocated by the dynamic allocation of ternary content addressable memory (TCAM) for firewall filters.
- On EX4300 switches, the following maximum number of terms are supported for ingress and egress traffic, for firewall filers configured on a port, VLAN and Layer 3 interface:
  - For ingress traffic:
    - 3500 terms for firewall filters configured on a port
    - 3500 terms for firewall filters configured on a VLAN
    - 7000 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
    - 3500 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic
  - For EX4300-MP devices, ingress support is the same as the above with the following exception:
    - 3072 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - For egress traffic:
    - 512 terms for firewall filters configured on a port
    - 256 terms for firewall filters configured on a VLAN
    - 512 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
    - 512 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic

NOTE: You can configure the maximum number of terms only when you configure one type of firewall filter (port, VLAN, or router (Layer 3) firewall filter) on the switch, and when storm control is not enabled on any interface in the switch.

- For EX4400 switches, the following maximum number of terms are supported for ingress and egress traffic, for firewall filters configured on a port, VLAN and layer 3 interfaces.
  - For ingress traffic:
    - 2048 terms for firewall filters configured on a port.

    - 2048 terms for firewall filters configured on a VLAN.
    - 2048 terms for firewall filters configured on layer 3 interfaces.
  - For egress traffic:
    - 1024 terms for firewall filters configured on a port.
    - 512 terms for firewall filters configured on a VLAN.
    - 1024 terms for firewall filters configured on layer 3 interfaces.
- 1200 for EX4500 and EX4550 switches
- 1400 for EX6200 switches
- 32,768 for EX8200 switches

NOTE: The on-demand dynamic allocation of the shared space TCAM in EX8200 switches is achieved by assigning free space blocks to firewall filters. Firewall filters are categorized into two different pools. Port and VLAN filters are pooled together (the memory threshold for this pool is 22K) while router firewall filters are pooled separately (the threshold for this pool is 32K). The assignment happens based on the filter pool type. Free space blocks can be shared only among the firewall filters belonging to the same filter pool type. An error message is generated when you try to configure a firewall filter beyond the TCAM threshold.

Each term consists of the following components:

- Match conditions—Specify the values or fields that the packet must contain. You can define various match conditions, including the IP source address field, IP destination address field, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) source port field, IP protocol field, Internet Control Message Protocol (ICMP) packet type, TCP flags, and interfaces.
- Action—Specifies what to do if a packet matches the match conditions. Possible actions are to accept or discard the packet or to send the packet to a specific virtual routing interface. In addition, packets can be counted to collect statistical information. If no action is specified for a term, the default action is to accept the packet.
- Action modifier—Specifies one or more actions for the switch if a packet matches the match conditions. You can specify action modifiers such as count, mirror, rate limit, and classify packets.

Firewall Filter Processing

The order of the terms within a firewall filter configuration is important. Packets are tested against each term in the order in which the terms are listed in the firewall filter configuration. For information on how firewall filters process packets, see "Understanding How Firewall Filters Are Evaluated" on page 1550.

RELATED DOCUMENTATION

Understanding Planning of Firewall Filters | **1539**

Understanding Firewall Filter Processing Points for Bridged and Routed Packets on EX Series Switches | **1552**

Understanding How Firewall Filters Are Evaluated | **1550**

Understanding Firewall Filter Match Conditions | **1543**

Understanding the Use of Policers in Firewall Filters | **2210**

Understanding Filter-Based Forwarding for EX Series Switches | **1650**

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**

Understanding Planning of Firewall Filters

Before you create a *firewall filter* and apply it to an interface, determine what you want the firewall filter to accomplish and how to use its match conditions and actions to achieve your goals. You must understand how packets are matched to match conditions, the default and configured actions of the firewall filter, and proper placement of the firewall filter.

You can configure and apply no more than one firewall filter per port, VLAN, or router interface, per direction. The following limits apply for the number of firewall filter terms allowed per filter on various switch models:

- On EX3300 switches, the number of terms per filter cannot exceed 1436.
- On EX3200 and EX4200 switches, the number of terms per filter cannot exceed 7042.
- On EX2300 switches, the following maximum number of terms are supported for ingress and egress traffic, for firewall filters configured on a port, VLAN and Layer 3 interface:
  - For ingress traffic:
    - 256 terms for firewall filters configured on a port

  - 256 terms for firewall filters configured on a VLAN
  - 256 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 256 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic
- For egress traffic:
  - 512 terms for firewall filters configured on a port
  - 128 terms for firewall filters configured on a VLAN
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic

On EX3400 switches, the following maximum number of terms are supported for ingress and egress traffic, for firewall filters configured on a port, VLAN and Layer 3 interface:

- For ingress traffic:
  - 512 terms for firewall filters configured on a port
  - 512 terms for firewall filters configured on a VLAN
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic
- For egress traffic:
  - 512 terms for firewall filters configured on a port
  - 256 terms for firewall filters configured on a VLAN
  - 1024 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 1024 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic

On EX4300 switches, the following maximum number of terms are supported for ingress and egress traffic, for firewall filers configured on a port, VLAN and Layer 3 interface:

- For ingress traffic:
  - 3500 terms for firewall filters configured on a port
  - 3500 terms for firewall filters configured on a VLAN
  - 7000 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 3500 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic

> **NOTE**: The ternary content addressable memory (TCAM) limit for ingress traffic on the EX4300 switch is 256 entries.

- For egress traffic:
  - 512 terms for firewall filters configured on a port
  - 256 terms for firewall filters configured on a VLAN
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
  - 512 terms for firewall filters configured on Layer 3 interfaces for IPv6 traffic

> **NOTE**: You can configure the maximum number of terms only when you configure one type of firewall filter (port, VLAN, or router (Layer 3) firewall filter) on the switch, and when storm control is not enabled on any interface in the switch.

- On EX4500 and EX4550 switches, the number of terms per filter cannot exceed 1200.
- On EX6200 switches, the number of terms per filter cannot exceed 1400.
- On EX8200 switches, the number of terms per filter cannot exceed 32,768.

In addition, try to be conservative in the number of terms (rules) that you include in each firewall filter because a large number of terms requires longer processing time during a commit and also can make firewall filter testing and troubleshooting more difficult. Similarly, applying firewall filters across many switch and router interfaces can make testing and troubleshooting the rules of those filters difficult.

Before you configure and apply firewall filters, answer the following questions for each of those firewall filters:

1. What is the purpose of the firewall filter?

   For example, you can use a firewall filter to limit traffic to source and destination MAC addresses, specific protocols, or certain data rates or to prevent denial of service (DoS) attacks.

2. What are the appropriate match conditions?

   a. Determine the packet header fields that the packet must contain for a match. Possible fields include:

      - Layer 2 header fields—Source and destination MAC addresses, dot1q tag, Ethernet type, and VLAN

- Layer 3 header fields—Source and destination IP addresses, protocols, and IP options (IP precedence, IP fragmentation flags, TTL type)
- TCP header fields—Source and destination ports and flags
- ICMP header fields—Packet type and code

b. Determine the port, VLAN, or router interface on which the packet was received.

3. What are the appropriate actions to take if a match occurs?

   Possible actions to take if a match occurs are accept, discard, and forward to a routing instance.

4. What additional action modifiers might be required?

   Determine whether additional actions are required if a packet matches a match condition; for example, you can specify an action modifier to count, analyze, or police packets.

5. On what interface should the firewall filter be applied?

   Start with the following basic guidelines:

   - If all the packets entering a port need to be exposed to filtering, then use port firewall filters.
   - If all the packets that are bridged need filtering, then use VLAN firewall filters.
   - If all the packets that are routed need filtering, then use router firewall filters.

   Before you choose the interface on which to apply a firewall filter, understand how that placement can impact traffic flow to other interfaces. In general, apply a firewall filter that filters on source and destination IP addresses, IP protocols, or protocol information—such as ICMP message types, and TCP and UDP port numbers—nearest to the source devices. However, typically apply a firewall filter that filters only on a source IP address nearest to the destination devices. When applied too close to the source device, a firewall filter that filters only on a source IP address could potentially prevent that source device from accessing other services that are available on the network.

   NOTE: Egress firewall filters do not affect the flow of locally generated control packets from the Routing Engine.

6. In which direction should the firewall filter be applied?

   You can apply firewall filters to ports on the switch to filter packets that are entering a port. You can apply firewall filters to VLANs, and Layer 3 (routed) interfaces to filter packets that are entering or exiting a VLAN or routed interface. Typically, you configure different sets of actions for traffic entering an interface than you configure for traffic exiting an interface.

RELATED DOCUMENTATION

- Firewall Filters for EX Series Switches Overview | 1535
- Understanding the Use of Policers in Firewall Filters | 2210
- Understanding How Firewall Filters Are Evaluated | 1550
- Understanding Filter-Based Forwarding for EX Series Switches | 1650
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650
- Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | 1685

Understanding Firewall Filter Match Conditions

IN THIS SECTION

- Filter Match Conditions | 1543
- Numeric Filter Match Conditions | 1544
- Interface Filter Match Conditions | 1545
- IP Address Filter Match Conditions | 1545
- MAC Address Filter Match Conditions | 1546
- Bit-Field Filter Match Conditions | 1547

Before you define terms for firewall filters, you must understand how the match conditions that you specify in a term are handled and how to specify various types of match conditions to achieve the desired filtering results. A match condition consists of a string (called a match statement) that defines the match condition. Match conditions are the values or fields that a packet must contain.

Filter Match Conditions

In the `from` statement of a *firewall filter* term, you specify the packet conditions that trigger the action in one of the `then` statements: **then** with various options, **then interface** or **then vlan**. All conditions in the `from` statement must match for the action to be taken. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify no match conditions in a term, that term matches all packets.

An individual condition in a `from` statement cannot contain a list of values. For example, you cannot specify numeric ranges or multiple source or destination addresses.

Individual conditions in a `from` statement cannot be negated. A negated condition is an explicit mismatch.

Numeric Filter Match Conditions

Numeric filter conditions match packet fields that are identified by a numeric value, such as port and protocol numbers. For numeric filter match conditions, you specify a keyword that identifies the condition and a single value that a field in a packet must match.

You can specify the numeric value 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;
```

- 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 http;
```

To specify more than one value in a filter term, you enter each value in its own match statement, which is a string that defines a match condition. For example, a match occurs in the following term if the value of **vlan** is 10 or 30.

```
[edit firewall family family-name filter filter-name term term-name from]
vlan 10;
vlan 30;
```

The following restrictions apply to numeric filter match conditions:

- You cannot specify a range of values.

- You cannot specify a list of comma-separated values.
- You cannot exclude a specific value in a numeric filter match condition. For example, you cannot specify a condition that would match only if the match condition was not equal to a given value.

Interface Filter Match Conditions

Interface filter match conditions can match interface name values in a packet. For interface filter match conditions, you specify the name of the interface, for example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/0/1
```

Port and VLAN interfaces do not use logical unit numbers. However, a firewall filter that is applied to a router interface can specify the logical unit number in the interface filter match condition, for example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/1/0.0
```

You can include the * wildcard as part of the interface name, for example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set interface ge-0/*/1
user@switch# set interface ge-0/1/*
user@switch# set interface ge-*
```

IP Address Filter Match Conditions

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

You specify the address as a 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]
user@switch# set destination-address 10.2.1.0/28;
```

Each prefix contains an implicit 0/0 except statement, which means that any prefix that does not match the prefix that is specified is explicitly considered not to match.

To specify the address prefix, use the notation prefix/prefix-length. If you omit prefix-length, it defaults to /32. For example:

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

To specify more than one IP address in a filter term, you enter each address in its own match statement. For example, a match occurs in the following term if the value of the **source-address** field matches either of the following source-address prefixes:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-address 10.0.0.0/8
user@switch# set source-address 10.1.0.0/16
```

MAC Address Filter Match Conditions

MAC address filter match conditions can match source and destination MAC address values in a packet. For MAC address filter match conditions, you specify a keyword that identifies the field and one value of that type that a packet must match.

You can specify the MAC address as six hexadecimal bytes in the following formats:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 0011.2233.4455
```

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 00:11:22:33:44:55
```

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set destination-mac-address 001122334455
```

To specify more than one MAC address in a filter term, you enter each MAC address in its own match statement. For example, a match occurs in the following term if the value of the **source-mac-address** field matches either of the following addresses.

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set source-mac-address 00:11:22:33:44:55
user@switch# set source-mac-address 00:11:22:33:20:15
```

Bit-Field Filter Match Conditions

Bit-field filter conditions match packet fields if particular bits in those fields are or are not set. You can match the IP options, TCP flags, and IP fragmentation fields. For bit-field filter match conditions, you specify a keyword that identifies the field and tests to determine that the option is present in the field.

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

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "rst"
```

Typically, you specify the bits to be tested by using keywords. Bit-field match keywords always map to a single bit value. You also can specify bit fields as hexadecimal or decimal numbers.

To match multiple bit-field values, use the logical operators, which are described in Table 74 on page 1547. The operators are listed in order from highest precedence to lowest precedence. Operations are left-associative.

Table 74: Logical Operators for Matching Multiple Bit-Field Operators

Logical Operators	Description
!	Negation.
&	Logical AND.
|	Logical OR.

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:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "!rst"
```

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

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn&!ack"
```

In the following example of a logical OR operation, a match occurs if the packet is part of TCP session establishment or tear down:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags "syn|fin"
```

For a logical OR operation, you can specify a maximum of two match conditions in a single term. If you need to match more than two bit-field values in a logical OR operation, configure the same match condition in consecutive terms with additional bit-field values. In the following example, the two terms configured match the SYN, ACK, FIN, or RST bit in the TCP flags field:

```
[edit firewall family family-name filter filter-name term term-name1 from]
user@switch# set tcp-flags "syn|ack"
[edit firewall family family-name filter filter-name term term-name2 from]
user@switch# set tcp-flags "fin|rst"
```

You can use text synonyms to specify some common bit-field matches. You specify these matches as a single keyword. In the following example of a text synonym, a match occurs if the packet is the initial packet on a TCP session:

```
[edit firewall family family-name filter filter-name term term-name from]
user@switch# set tcp-flags tcp-initial
```

RELATED DOCUMENTATION

Firewall Filters for EX Series Switches Overview | 1535

Understanding How Firewall Filters Test a Packet's Protocol | 1649

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | 1685

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | 1554

Understanding How Firewall Filters Control Packet Flows

Juniper Networks EX Series Ethernet Switches support firewall filters that allow you to control flows of data packets and local packets. *Data packets* are chunks of data that transit the switch as they are forwarded from a source to a destination. *Local packets* are chunks of data that are destined for or sent by the switch. 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).

You create firewall filters to protect your switch from excessive traffic transiting the switch to a network destination or destined for the Routing Engine on the switch. Firewall filters that control local packets can also protect your switch from external incidents such as denial-of-service (DoS) attacks.

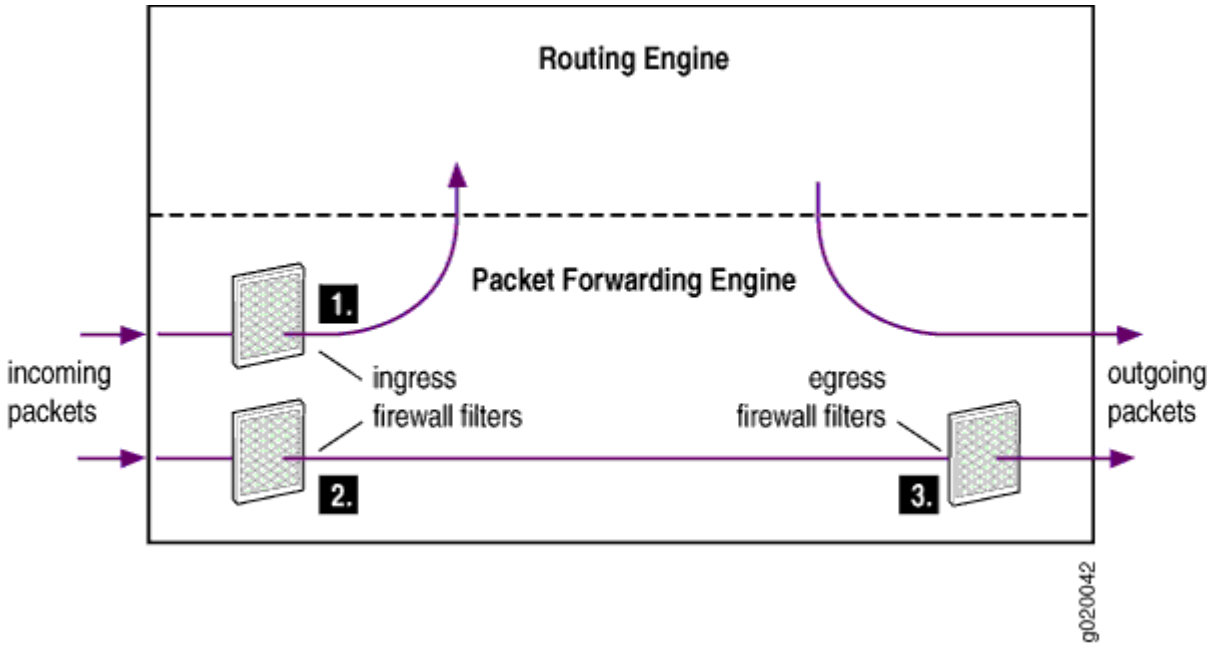
Firewall filters affect packet flows entering in to or exiting from the switch's interfaces:

- Ingress firewall filters affect the flow of data packets that are received by the switch's interfaces. The Packet Forwarding Engine handles this flow. When a switch receives a data packet on an interface, the switch determines where to forward the packet by looking in the forwarding table for the best route (Layer 2 switching, Layer 3 routing) to a destination. Data packets are forwarded to their destination through an outgoing interface. Locally destined packets are forwarded to the Routing Engine.

- Egress firewall filters affect the flow of data packets that are transmitted from the switch's interfaces but do not affect the flow of locally generated control packets from the Routing Engine. The Packet Forwarding Engine handles the flow of data packets that are transmitted from the switch, and egress firewall filters are applied here. The Packet Forwarding Engine also handles the flow of control packets from the Routing Engine.

Figure 69 on page 1550 illustrates the application of ingress and egress firewall filters to control the flow of packets through the switch.

Figure 69: Application of Firewall Filters to Control Packet Flow



1. Ingress *firewall filter* applied to control locally destined packets that are received on the switch's interfaces and are destined for the Routing Engine.
2. Ingress firewall filter applied to control incoming packets on the switch's interfaces.
3. Egress firewall filter applied to control packets that are transiting the switch's interfaces.

RELATED DOCUMENTATION

Understanding Firewall Filter Processing Points for Bridged and Routed Packets on EX Series Switches | **1552**

Understanding How Firewall Filters Are Evaluated | **1550**

Understanding How Firewall Filters Are Evaluated

A *firewall filter* consists of one or more terms, and the order of the terms within a firewall filter is important. Before you configure firewall filters, you should understand how Juniper Networks EX Series Ethernet Switches evaluate the terms within a firewall filter and how packets are evaluated against the terms.

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 no action is specified in the `then` statement, the default action **accept** is taken.

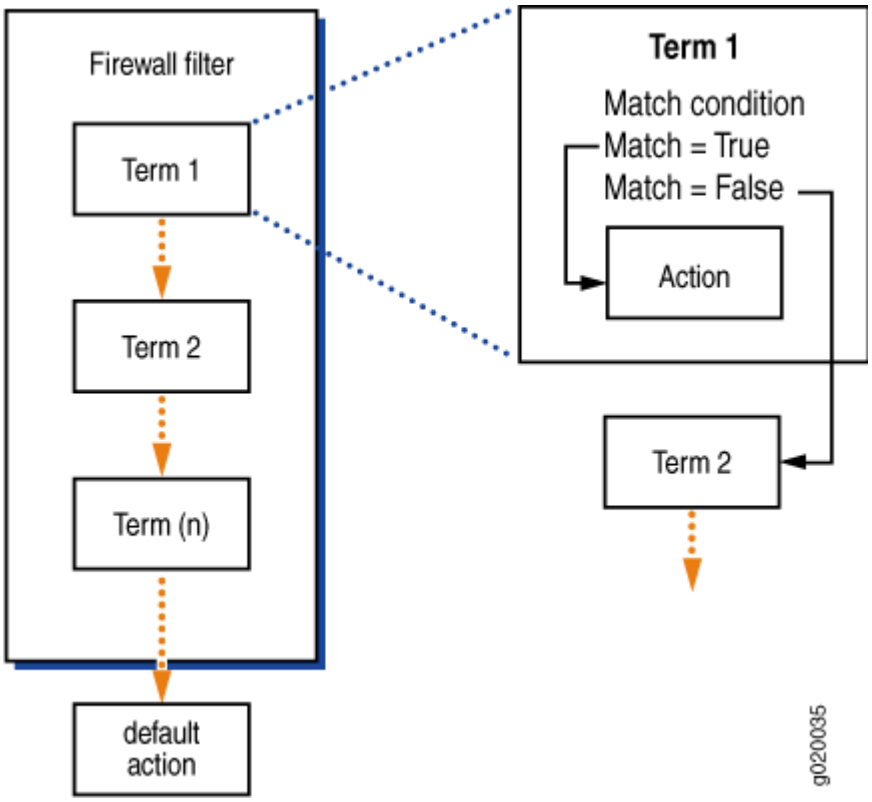
When a firewall filter consists of more than one term, the firewall filter is evaluated sequentially:

1. The packet is evaluated against the conditions in the `from` statement in the first term.
2. If the packet matches all the conditions in the term, the action in the `then` statement is taken and the evaluation ends. Subsequent terms in the filter are not evaluated.
3. If the packet does not match all the conditions in the term, the packet is evaluated against the conditions in the `from` statement in the second term.

   This process continues until either the packet matches the conditions in the `from` statement in one of the subsequent terms or there are no more terms in the filter.
4. If a packet passes through all the terms in the filter without a match, the packet is discarded.

Figure 70 on page 1551 shows how an EX Series switch evaluates the terms within a firewall filter.

Figure 70: Evaluation of Terms Within a Firewall Filter



If a term does not contain a `from` statement, the packet is considered to match and the action in the `then` statement of the term is taken.

If a term does not contain a `then` statement, or if an action has not been configured in the `then` statement, and the packet matches the conditions in the `from` statement of the term, the packet is accepted.

Every firewall filter contains an implicit `deny` statement at the end of the filter, which is equivalent to the following explicit filter term:

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

Consequently, if a packet passes through all the terms in a filter without matching any conditions, the packet is discarded. If you configure a firewall filter that has no terms, all packets that pass through the filter are discarded.

RELATED DOCUMENTATION

- Firewall Filters for EX Series Switches Overview | **1535**
- Understanding Firewall Filter Match Conditions | **1543**
- Understanding the Use of Policers in Firewall Filters | **2210**
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

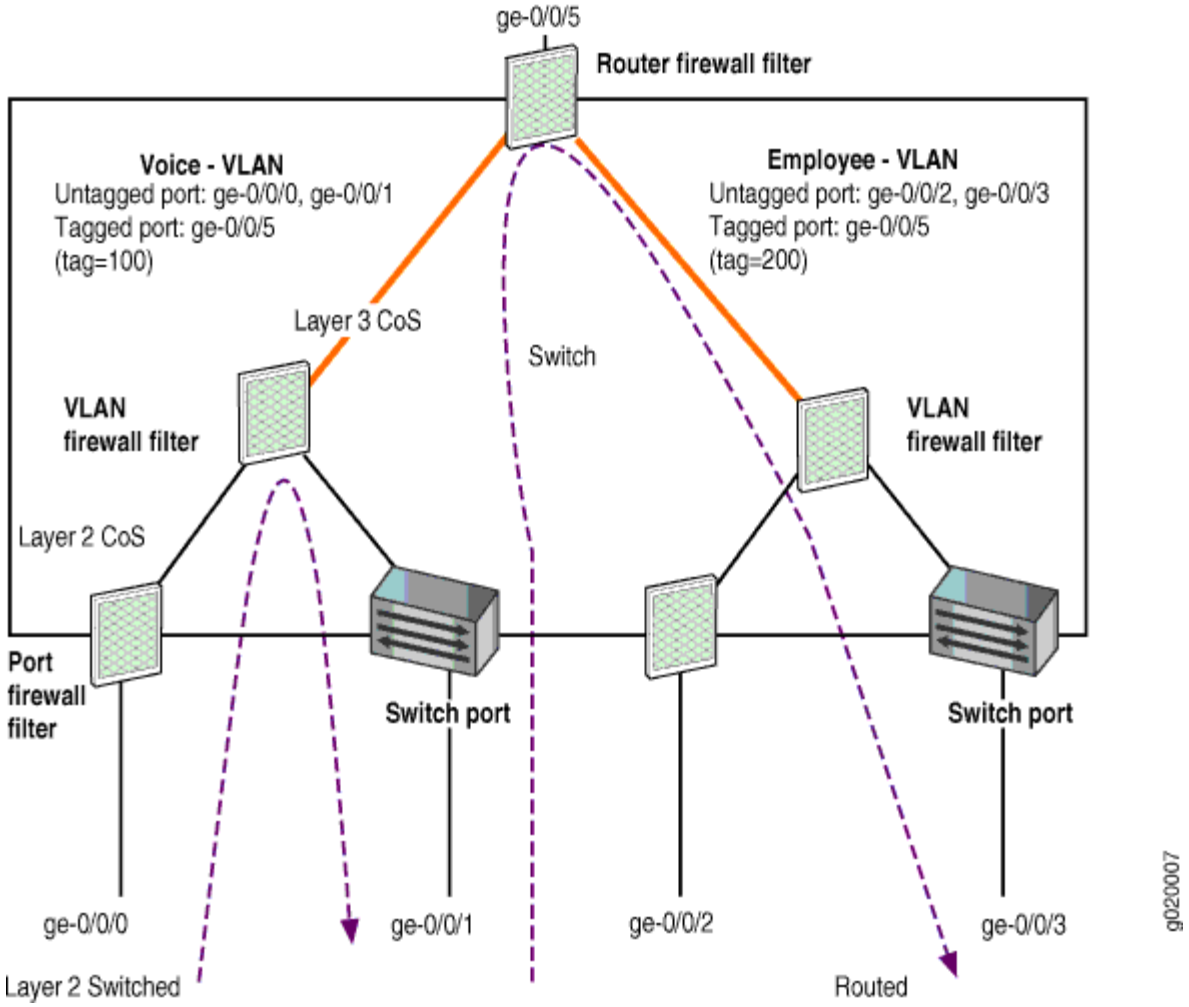
Understanding Firewall Filter Processing Points for Bridged and Routed Packets on EX Series Switches

Juniper Networks EX Series Ethernet Switches are multilayered switches that provide Layer 2 switching and Layer 3 routing. You apply firewall filters at multiple processing points in the packet forwarding path on EX Series switches. At each processing point, the action to be taken on a packet is determined based on the results of the lookup in the switch's forwarding table. A table lookup determines which exit port on the switch to use to forward the packet.

For both bridged unicast packets and routed unicast packets, firewall filters are evaluated and applied hierarchically. First, a packet is checked against the port *firewall filter*, if present. If the packet is permitted, it is then checked against the VLAN firewall filter, if present. If the packet is permitted, it is then checked against the router firewall filter, if present. The packet must be permitted by the router firewall filter before it is processed.

Figure 71 on page 1553 shows the various firewall filter processing points in the packet forwarding path in a multilayered switching platform.

Figure 71: Firewall Filter Processing Points in the Packet Forwarding Path



For a multicast packet that results in replications, an egress firewall filter is applied to each copy of the packet based on its corresponding egress VLAN.

For Layer 2 (bridged) unicast packets, the following firewall filter processing points apply:

- Ingress port firewall filter
- Ingress VLAN firewall filter
- Egress port firewall filter
- Egress VLAN firewall filter

For Layer 3 (routed and multilayer-switched) unicast packets, the following firewall filter processing points apply:

- Ingress port firewall filter
- Ingress VLAN firewall filter (Layer 2 CoS)
- Ingress router firewall filter (Layer 3 CoS)
- Egress router firewall filter
- Egress VLAN firewall filter

RELATED DOCUMENTATION

Firewall Filters for EX Series Switches Overview | 1535

Understanding How Firewall Filters Control Packet Flows | 1549

Understanding Bridging and VLANs on Switches

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches

IN THIS SECTION

- Firewall Filter Elements | 1555
- Match Conditions Supported on Switches | 1555
- Actions for Firewall Filters | 1566
- Action Modifiers for Firewall Filters | 1567

When you define a firewall filter for an EX Series switch, you define filtering criteria (*terms*, with *match conditions*) for the packets and an *action* (and, optionally, an *action modifier*) for the switch to take if the packets match the filtering criteria. You can define a firewall filter to monitor IPv4, IPv6, or non-IP traffic.

This topic describes in detail the various match conditions, actions, and action modifiers that you can define in a firewall filter. For information about support for match conditions on various EX Series

switches, see "Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches" on page 1570.

Firewall Filter Elements

A firewall filter configuration contains a term, a match condition, an action, and, optionally, an action modifier. Table 75 on page 1555 describes each element in a firewall filter configuration.

Table 75: Elements of a Firewall Filter Configuration

Element Name	Description
Term	Defines the filtering criteria for the packets. Each term in the firewall filter consists of match conditions and an action. You can define a single term or multiple terms in the firewall filter. If you define multiple terms, each term must have a unique name.
Match condition	Consists of a string (called a *match statement*) that defines the match condition. Match conditions are the values or fields that a packet must contain. You can define a single match condition or multiple match conditions for a term. You can also opt not to define a match condition. If no match conditions are specified for a term, all packets are matched by default.
Action	Specifies the action that the switch takes if a packet matches all the criteria specified in the match conditions.
Action modifier	Specifies one or more actions that the switch takes if a packet matches the match conditions for the specific term.

Match Conditions Supported on Switches

Based on the type of traffic that you want to monitor, you can configure a firewall filter to monitor IPv4, IPv6, or non-IP traffic. When you configure a firewall filter to monitor a particular type of traffic, ensure that you specify match conditions that are supported for that type of traffic. For information about match conditions supported for a specific type of traffic and switches on which they are supported, see "Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches" on page 1570.

Table 76 on page 1556 describes all the match conditions that are supported for firewall filters on EX Series Switches.

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches

Match Condition	Description
`destination-address` *ip-address*	IP destination address field, which is the address of the final destination node.
`ip-destination-address` *ip-address*	IP destination address field, which is the address of the final destination node.
`ip6-destination-address` *ip-address*	IP destination address field, which is the address of the final destination node.
destination-mac-address *mac-address*	Destination media access control (MAC) address of the packet. You can define a destination MAC address with a prefix, such as **destination-mac-address 00:01:02:03:04:05/24**. If no prefix is specified, the default value 48 is used.
destination-port *number*	TCP or UDP destination port field. Typically, you specify this match condition in conjunction with the `protocol` or **ip-protocol** match condition to determine which protocol is used on the port. For *number*, 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)**, **nntp (119)**, **ntalk (518)**, **ntp (123)**, **pop3 (110)**, **pptp (1723)**, **printer (515)**, **radacct (1813)**,**radius (1812)**, **rip (520)**, **rkinit (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)**, **xdmcp (177)**, **zephyr-clt (2103)**, **zephyr-hm (2104)**

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
destination-prefix-list *prefix-list*	IP destination prefix list field. You can define a list of IP address prefixes under a prefix-list alias for frequent use. You define this match condition at the `[edit policy-options]` hierarchy level.
`dot1q-tag` *number*	The tag field in the Ethernet header. The tag values range from 1 through 4095. The `dot1q-tag` match condition and the **vlan** match condition are mutually exclusive.
`user-vlan-id` *number*	The tag field in the Ethernet header. The tag values range from 1 through 4095. The `user-vlan-id` match condition and the `learn-vlan-id` match condition are mutually exclusive.
dot1q-user-priority *number*	User-priority field of the tagged Ethernet packet. User-priority values can range from 0 through 7. For *number*, you can specify one of the following text synonyms (the field values are also listed): • **background (1)**—Background • **best-effort (0)**—Best effort • **controlled-load (4)**—Controlled load • **excellent-load (3)**—Excellent load • **network-control (7)**—Network control reserved traffic • **standard (2)**—Standard or spare • **video (5)**—Video • **voice (6)**—Voice

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
user-vlan-1p-priority *number*	User-priority field of the tagged Ethernet packet. User-priority values can range from 0 through 7. For *number*, you can specify one of the following text synonyms (the field values are also listed): • **background (1)**—Background • **best-effort (0)**—Best effort • **controlled-load (4)**—Controlled load • **excellent-load (3)**—Excellent load • **network-control (7)**—Network control reserved traffic • **standard (2)**—Standard or spare • **video (5)**—Video • **voice (6)**—Voice
dscp *number*	Specifies the Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant six bits of this byte form the DSCP. You can specify DSCP in hexadecimal, binary, or decimal form. For *number*, you can specify one of the following text synonyms (the field values are also listed): • **ef (46)**—as defined in RFC 2598, *An Expedited Forwarding PHB*. • **af11 (10)**, **af12 (12)**, **af13 (14)**, **af21 (18)**, **af22 (20)**, **af23 (22)**, **af31 (26)**, **af32 (28)**, **af33 (30)**, **af41 (34)**, **af42 (36)**, **af43 (38)** These four classes, with three drop precedences in each class, are defined for 12 code points in RFC 2597, *Assured Forwarding PHB Group*.

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
ether-type *value*	Ethernet type field of a packet. The *value* specifies what protocol is being transported in the Ethernet frame. For *value*, you can specify one of the following text synonyms: • **aarp**—EtherType value AARP (0x80F3) • **appletalk**—EtherType value AppleTalk (0x809B) • **arp**—EtherType value ARP (0x0806) • **ipv4**—EtherType value IPv4 (0x0800) • **ipv6**—EtherType value IPv6 (0x08DD) • **mpls multicast**—EtherType value MPLS multicast (0x8848) • **mpls unicast**—EtherType value MPLS unicast (0x8847) • **oam**—EtherType value OAM (0x88A8) • **ppp**—EtherType value PPP (0x880B) • **pppoe-discovery**—EtherType value PPPoE Discovery Stage (0x8863) • **pppoe-session**—EtherType value PPPoE Session Stage (0x8864) • **sna**—EtherType value SNA (0x80D5) **NOTE**: The following match conditions are not supported when **ether-type** is set to **ipv6**: • **dscp** • **fragment-flags** • **is-fragment** • `precedence` or `ip-precedence` • `protocol` or `ip-protocol`

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
fragment-flags ***fragment-flags***	IP fragmentation flags, specified in symbolic or hexadecimal formats. You can specify one of the following options: • **dont-fragment** (0x4000) • **more-fragments** (0x2000) • **reserved** (0x8000)
gbp-dst-tag	Match the destination tag, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN
gbp-src-tag	Match the source tag, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN
icmp-code ***number***	ICMP code field. This value or option 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 *number*, you can specify one of the following text synonyms (the field values are also listed). The options are grouped by the ICMP type with which they are associated: • **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)**

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
icmp-type *number*	ICMP packet type field. Typically, you specify this match condition in conjunction with the `protocol` or `ip-protocol` match condition to determine which protocol is being used on the port. For *number*, you can specify one of the following text synonyms (the field values are also listed): **echo-reply (0)**, **echo-request (8)**, **info-reply (16)**, **info-request (15)**, **mask-request (17)**, **mask-reply (18)**, **parameter-problem (12)**, **redirect (5)**, **router-advertisement (9)**, **router-solicit (10)**, **source-quench (4)**, **time-exceeded (11)**, **timestamp (13)**, **timestamp-reply (14)**, **unreachable (3)**
interface *interface-name*	Interface on which the packet is received. You can specify the wildcard character (*) as part of an interface name. **NOTE**: The **interface** match condition is not supported for egress traffic on an EX8200 Virtual Chassis.
ip-options	Presence of the options field in the IP header.

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
ip-version *version match_condition(s)*	Version of the IP protocol for port and VLAN firewall filters. The value for *version* can be **ipv4** or **ipv6**. For *match_condition(s)*, you can specify one or more of the following match conditions: • `destination-address`, `ip-destination-address`, or `ip6-destination-address` • **destination-port** • **destination-prefix-list** • **dscp** • **fragment-flags** • **icmp-code** • **icmp-type** • **is-fragment** • `precedence` or `ip-precedence` • `protocol` or `ip-protocol` • `source-address` or `ip-source-address` • **source-port** • **source-prefix-list** • **tcp-established** • **tcp-flags** • **tcp-initial**

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
is-fragment	If the packet is a trailing fragment, this match condition does not match the first fragment of a fragmented packet. Use two terms to match both first and trailing fragments. **NOTE**: Due to a limitation on the EX2300, EX3400, and EX4300 switches, this match condition does not match the last fragment of a fragmented packet when applied to "family ethernet-switching."
l2-encap-type llc-non-snap	Match on logical link control (LLC) layer packets for non-Subnet Access Protocol (SNAP) Ethernet Encapsulation type.
next-header *bytes*	8-bit 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): **ah (51)**, **dstops (60)**, **egp (8)**, **esp (50)**, **fragment (44)**, **gre (47)**, **hop-by-hop (0)**, **icmp (1)**, **icmp6 (1)**, **igmp (2)**, **ipip (4)**, **ipv6 (41)**, **no-next-header (59)**, **ospf (89)**, **pim (103)**, **routing (43)**, **rsvp (46)**, **sctp (132)**, **tcp (6)**, **udp (17)**, **vrrp (112)**
packet-length *bytes*	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.
`precedence` *precedence*	IP precedence. For *precedence*, you can specify one of the following text synonyms (the field values are also listed): **critical-ecp (5)**, **flash (3)**, **flash-override (4)**, **immediate (2)**, **internet-control (6)**, **net-control (7)**, **priority (1)**, **routine (0)**
`ip-precedence` *precedence*	IP precedence. For *precedence*, you can specify one of the following text synonyms (the field values are also listed): **critical-ecp (5)**, **flash (3)**, **flash-override (4)**, **immediate (2)**, **internet-control (6)**, **net-control (7)**, **priority (1)**, **routine (0)**

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
`protocol` *`list of protocol`*	IPv4 protocol value. For *protocols*, you can specify one of the following text synonyms: **egp (8)**, **esp (50)**, **gre (47)**, **icmp (1)**, **igmp (2)**, **ipip (4)**, **ospf (89)**, **pim (103)**, **rsvp (46)**, **tcp (6)**, **udp (17)**
`ip-protocol` *`list of protocol`*	IPv4 protocol value. For *protocols*, you can specify one of the following text synonyms: **egp (8)**, **esp (50)**, **gre (47)**, **icmp (1)**, **igmp (2)**, **ipip (4)**, **ospf (89)**, **pim (103)**, **rsvp (46)**, **tcp (6)**, **udp (17)**
source-address ***ip-address***	IP source address field, which is the address of the source node sending the packet. For IPv6, the source-address field is 128 bits in length. The filter description syntax supports the text representations for IPv6 addresses that are described in RFC 2373, *IP Version 6 Addressing Architecture*.
`ip-source-address (`*`ip-address`* `|` *`ip6-address`*`)`	IP source address field, which is the address of the source node sending the packet. You can specify either an IPv4 address (`ip-address`) or an IPv6 address (`ip6-address`). For IPv6, the ip-source-address field is 128 bits in length. The filter description syntax supports the text representations for IPv6 addresses that are described in RFC 2373, *IP Version 6 Addressing Architecture*.
source-mac-address ***mac-address***	Source MAC address. You can define a source MAC address with a prefix, such as **source-mac-address 00:01:02:03:04:05/24**. If no prefix is specified, the default value 48 is used.
source-port ***number***	TCP or UDP **source-port** field. Typically, you specify this match in conjunction with the `protocol` or **ip-protocol** match condition to determine which protocol is being used on the port. For *number*, you can specify one of the text synonyms listed under **destination-port**.

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
source-prefix-list ***prefix-list***	IP source prefix list field. You can define a list of IP address prefixes under a prefix-list alias for frequent use. You define this match condition at the `[edit policy-options]` hierarchy level.
tcp-established	TCP packets of an established TCP connection. This condition matches packets other than the first packet of a connection. **tcp-established** is a synonym for the bit names **"(ack | rst)"**. **tcp-established** does not implicitly check whether the protocol is TCP. To do so, specify the **next-header tcp** match condition.
tcp-flags (*flags* tcp-initial)	One or more TCP flags: • bit-name—**fin**, **syn**, **rst**, **push**, **ack**, **urgent** • logical operators—**&** (logical AND), **|** (logical OR), **!** (negation) • numerical value—0x01 through 0x20 • text synonym—**tcp-initial** To specify multiple flags, use logical operators.
tcp-initial	Matches the first TCP packet of a connection. **tcp-initial** is a synonym for the bit names **"(syn&!ack)"**. **tcp-initial** does not implicitly check whether the protocol is TCP. To do so, specify the `protocol tcp` or **ip-protocol tcp** match condition.
traffic-class ***number***	Specifies the DSCP code point for a packet.
ttl ***value***	TTL type to match. The value ranges from 1 through 255.
`vlan (`*`vlan-name`*` | `*`vlan-id`*`)`	The VLAN that is associated with the packet. For *vlan-id*, you can specify either the VLAN ID or a VLAN range. The **vlan** match condition and the **dot1q-tag** match condition are mutually exclusive.

Table 76: Firewall Filter Match Conditions Supported on EX Series Switches *(Continued)*

Match Condition	Description
`learn-vlan-id (vlan-name | vlan-id)`	The VLAN that is associated with the packet. For *vlan-id*, you can specify either the VLAN ID or a VLAN range. The **vlan** match condition and the **user-vlan-id** match condition are mutually exclusive.

Actions for Firewall Filters

You can define an action for the switch to take if a packet matches the filtering criteria defined in a match condition. Table 77 on page 1566 describes the actions supported in a firewall filter configuration.

Table 77: Actions for Firewall Filters

Action	Description
accept	Accept a packet.
discard	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.
reject ***message-type***	Discard a packet, and send the ICMPv4 message (type 3) **destination unreachable**. You can log the rejected packets if you configure 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, tcp-reset**. If you specify **tcp-reset**, a TCP reset is returned if the packet is a TCP packet. Otherwise nothing is returned. If you do not specify a message type, the ICMP notification **destination unreachable** is sent with the default message **communication administratively filtered**.

Table 77: Actions for Firewall Filters *(Continued)*

Action	Description
routing-instance ***routing-instance-name***	Forward matched packets to a virtual routing instance. **NOTE**: EX4200 switches do not support firewall-filter-based redirection to the default routing instance.
vlan ***vlan-name***	Forward matched packets to a specific VLAN. Ensure that you specify the VLAN name or VLAN ID and not a VLAN range, because the **vlan** action does not support the *vlan-range* option. **NOTE**: If you have defined a VLAN that is enabled for dot1q tunneling, then that particular VLAN is not supported as an action (using the **vlan** ***vlan-name*** action) for an ingress VLAN firewall filter.

Action Modifiers for Firewall Filters

In addition to the actions described in Table 77 on page 1566, you can define action modifiers in a firewall filter configuration for a switch if packets match the filtering criteria defined in the match condition. Table 78 on page 1567 describes the action modifiers supported in a firewall filter configuration.

Table 78: Action Modifiers for Firewall Filters

Action Modifier	Description
analyzer ***analyzer-name***	Mirror port traffic to a specified destination port or VLAN that is connected to a protocol analyzer application. Mirroring copies all packets seen on one switch port to a network monitoring connection on another switch port. The analyzer name must be configured under `[edit ethernet-switching-options analyzer]`. **NOTE**: **analyzer** is not a supported action modifier for a management interface. **NOTE**: On EX4500 switches, you can configure only one analyzer and include it in a firewall filter. If you configure multiple analyzers, you cannot include any one of those analyzers in a firewall filter.

Table 78: Action Modifiers for Firewall Filters *(Continued)*

Action Modifier	Description
dscp *number*	Change the DSCP value for matched packets to the DSCP value specified with this action modifier. *number* specifies the Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant six bits of this byte form the DSCP. You can specify DSCP in hexadecimal, binary, or decimal form. For *number*, you can specify one of the following text synonyms (the field values are also listed): • **ef (46)**—as defined in RFC 2598, *An Expedited Forwarding PHB*. • **af11 (10)**, **af12 (12)**, **af13 (14)**, **af21 (18)**, **af22 (20)**, **af23 (22)**, **af31 (26)**, **af32 (28)**, **af33 (30)**, **af41 (34)**, **af42 (36)**, **af43 (38)** These four classes, with three drop precedences in each class, are defined for 12 code points in RFC 2597, *Assured Forwarding PHB Group*.
count *counter-name*	Count the number of packets that pass this filter, term, or policer. A policer enables you to specify rate limits on traffic that enters an interface on a switch. **NOTE**: On EX4300 switches, you can configure the same number of counters and policers as the number of terms in the ternary content addressable memory (TCAM).
forwarding-class *class*	Classify the packet in one of the following forwarding classes: • **assured-forwarding** • **best-effort** • **expedited-forwarding** • **network-control**
gbp-src-tag (EX4400 and EX4650 only)	Set the group based policy source tag (0..65535) for use with micro-segmentation on VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN

Table 78: Action Modifiers for Firewall Filters *(Continued)*

Action Modifier	Description
interface ***interface-name***	Forward the traffic to the specified interface bypassing the switching lookup.
log	Log the packet's header information in the Routing Engine. To view this information, issue the `show firewall log` command in the CLI. **NOTE**: If the **log** or the **syslog** action modifier is configured along with a **vlan** action or an **interface** action modifier, the events might not be logged. However, the redirect interface functionality works as expected.
loss-priority (high | low)	Set the packet loss priority (PLP).
policer ***policer-name***	Apply rate limits to the traffic. You can specify a policer in a firewall filter only for ingress traffic on a port, VLAN, and router. **NOTE**: A counter for a policer is not supported on EX8200 switches. **NOTE**: On EX4300 switches, you can configure the same number of counters and policers as the number of terms in the TCAM.
`port-mirror`	Mirror packets to the interface defined in the `[edit forwarding-options analyzer]` hierarchy.
`port-mirror-instance` ***instance-name***	Mirror packets to the instance defined in the `[edit forwarding-options analyzer]` hierarchy.
syslog	Log an alert for this packet. You can specify that the log be sent to a server for storage and analysis. **NOTE**: If the **log** or the **syslog** action modifier is configured along with a **vlan** action or an **interface** action modifier, the events might not be logged. However, the redirect interface functionality works as expected.
three-color-policer	Apply a three-color policer.

RELATED DOCUMENTATION

Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches | 1570

Understanding Firewall Filter Match Conditions | 1543

Firewall Filter Configuration Statements Supported by Junos OS for EX Series Switches | 2254

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | 1685

Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches

IN THIS SECTION

- Firewall Filter Types and Their Bind Points | 1571
- Support for IPv4 and IPv6 Firewall Filters on Switches | 1571
- Platform Support for Match Conditions for IPv4 Traffic | 1572
- Platform Support for Match Conditions for IPv6 Traffic | 1592
- Platform Support for Match Conditions for Non-IP Traffic | 1606
- Platform Support for Actions for IPv4 Traffic | 1607
- Platform Support for Actions for IPv6 Traffic | 1611
- Platform Support for Action Modifiers for IPv4 Traffic | 1615
- Platform Support for Action Modifiers for IPv6 Traffic | 1624

After you define a firewall filter on an EX Series switch, you must associate the filter to a bind point so that the filter can filter the packets that enter or exit the bind point. Port firewall filters, VLAN firewall filters, and Layer 3 (or router) firewall filters are the different types of firewall filters you can apply on a switch, depending on the bind points the filters are associated with. While a port firewall filter applies to Layer 2 interfaces, a VLAN firewall filter applies to packets that enter or leave a VLAN and also to packets that are bridged within a VLAN. A Layer 3 firewall filter applies to Layer 3 (routed) interfaces and routed VLAN interfaces (RVIs).

NOTE: If you want to control the traffic that enters the Routing Engine of the switch, you must configure a firewall filter on the loopback interface (lo0) of the switch. For information about match conditions, actions, and action modifiers supported on the loopback interface of a switch, see "Support for Match Conditions and Actions for Loopback Firewall Filters on Switches" on page 1633.

This topic describes the supported switches and bind points for match conditions, actions, and action modifiers for firewall filters supported on EX Series switches. For descriptions of the match conditions, actions, and action modifiers, see "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554. For information about the EX4600 switch, see Firewall Filter Match Conditions and Actions.

Firewall Filter Types and Their Bind Points

You can apply a firewall filter at specific bind points to filter IPv4, IPv6, or non-IP traffic. See the remaining sections in this topic for information about support on individual switches for different traffic types.

Table 79 on page 1571 lists the firewall filter types and their associated bind points that are supported on the switches.

Table 79: Bind Points Associated with Firewall Filter Types

Bind Points	Firewall Filter Type
Ports (Layer 2 interfaces)	Port firewall filter
VLANs	VLAN firewall filter
Layer 3 interfaces (Layer 3 (routed) interfaces or routed VLAN interfaces (RVIs)	Router firewall filter

Support for IPv4 and IPv6 Firewall Filters on Switches

On EX2200, EX2300/EX3400, EX3200/EX4200, EX3300, EX4500, and EX6200 switches port and VLAN filters on IPv6 traffic can match only layer 2 header fields. On an EX8200 switch, port and VLAN

traffic can match on layer 3 and layer 4 header fields, in addition to layer 2 header fields, of IPv6 traffic. On an EX4300 switch, port and VLAN filters on IPv6 traffic can match layer 3 and layer 4 header fields.

Table 80 on page 1572 briefly summarizes the support for IPv4 and IPv6 firewall filters on different switches. The support for port, VLAN, and router firewall filters on different switches is further discussed in the subsequent sections in this topic.

Table 80: Support for IPv4 and IPv6 Firewall Filters on Switches

Switch	Support for IPv4 Firewall Filter	Support for IPv6 Firewall Filter
EX2200	Yes	Yes
EX2300 and EX3400	Yes	Yes
EX3200 and EX4200	Yes	Yes
EX3300	Yes	Yes
EX4300	Yes	Yes
EX4500	Yes	Yes
EX6200	Yes	Yes
EX8200	Yes	Yes

Platform Support for Match Conditions for IPv4 Traffic

You can define port, VLAN, and router firewall filters for ingress and egress IPv4 traffic on all EX Series switches. Table 81 on page 1573 summarizes the support for match conditions on different bind points for ingress and egress IPv4 traffic on different switches.

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`destination-address` *ip-address*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-destination-address`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Not supported (EX2300) Ports and VLANs (EX3400)
`destination-mac-address` *mac-address*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`destination-port` *number*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces (EX4300) Not supported (EX2300)
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`destination-prefix-list` *`prefix-list`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`dot1q-tag` *number*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Not supported
`user-vlan-id number`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
`dot1q-user-priority` *number*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`user-vlan-1p-priority number`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
`dscp` *`number`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ether-type` *value*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Not supported
`fragment-flags` *fragment-flags*	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`icmp-code` *number*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 Interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`icmp-type` *number*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 Interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`interface` *interface-name* **NOTE**: This match condition is not supported by firewall filters configured on ingress L3 interfaces and ingress VLAN interfaces when the interface to be matched is aggregate Ethernet (ae) interface.	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-options`	EX2200	Layer 3 interfaces	Not supported

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Layer 3 interfaces	Not supported
`ip-version` *versionmatch_condition(s)*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`is-fragment` **NOTE**: Due to a limitation on the EX2300, EX3400, and EX4300 switches, this match condition does not match the last fragment of a fragmented packet when applied to a port or a VLAN.	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`precedence` *precedence*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-precedence precedence`	EX4300	Ports and VLANs	Not supported
	EX2300 and EX3400	Ports and VLANs	Not supported
`protocol` *`list of protocols`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-protocol list of protocols`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
`source-address` *`ip-address`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Layer 3 interfaces	Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-source-address ip-address`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Not supported (EX2300) Ports and VLANs (EX3400)
`source-mac-address` *`mac-address`*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`source-port` *number*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`source-prefix-list` *prefix-list*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`tcp-established`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`tcp-flags (`*`flags`* `tcp-initial)`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`tcp-initial`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`ttl` *value*	EX2200	Layer 3 interfaces	Not supported
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Not supported
	EX3300	Layer 3 interfaces	Not supported
	EX4300	Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Not supported
	EX8200	Layer 3 interfaces	Not supported
`vlan` (*vlan-name* | *vlan-id*)	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported

Table 81: Firewall Filter Match Conditions Supported for IPv4 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports and VLANs	Ports
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`learn-vlan-id vlan-id`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported

Platform Support for Match Conditions for IPv6 Traffic

Table 82 on page 1592 summarizes support for match conditions on different bind points for ingress and egress IPv6 traffic on different switches.

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`destination-address` *ip-address*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 (routed) interfaces only
	EX3300	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Layer 3 interfaces	Layer 3 (routed) interfaces only
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces
`destination-mac-address` *mac-address*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`destination-port` *number*	EX2200	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`destination-prefix-list` *`prefix-list`*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 (routed) interfaces only
	EX3300	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`dot1q-tag` *`number`*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Not supported
`user-vlan-id number`	EX4300	Ports and VLANs	Ports and VLANs

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`dot1q-user-priority` *number*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`user-vlan-1p-priority number`	EX4300	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Not supported
`ether-type (ipv6)` *value*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Ports and VLANs	Ports and VLANs
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`icmp-code` *number*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`icmp-type` *number*	EX2200	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`interface` *`interface-name`* **NOTE**: This match condition is not supported by firewall filters configured on ingress L3 interfaces and ingress VLAN interfaces when the interface to be matched is	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
aggregate Ethernet (ae) interface.	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`ip-version` *version match_condition(s)*	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Not supported	Not supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported
	EX8200	Ports and VLANs	Ports and VLANs
`next-header` *bytes*	EX2200	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`packet-length` *bytes*	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Not supported	Not supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Layer 3 interfaces	Not supported
`source-address` *ip-address*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Not supported
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`source-mac-address` *mac-address*	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Ports and VLANs	Ports and VLANs

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs
`source-port` *number*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, Layer 3 interfaces	Ports and VLANs (EX3400) Not supported (EX2300)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`source-prefix-list` *prefix-list*	EX2200	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`tcp-established`	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`tcp-flags (flags tcp-initial)`	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`tcp-initial`	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports and VLANs (EX3400)
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports and VLANs
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`traffic-class` *number*	EX2200	Layer 3 interfaces	Layer 3 interfaces
	EX2300 and EX3400	Layer 3 interfaces	Layer 3 interfaces
	EX3200 and EX4200	Layer 3 interfaces	Layer 3 interfaces
	EX3300	Layer 3 interfaces	Layer 3 interfaces

Table 82: Firewall Filter Match Conditions Supported for IPv6 Traffic on Switches *(Continued)*

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
	EX4300	Layer 3 interfaces	Layer 3 interfaces
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Layer 3 interfaces	Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`vlan (`*`vlan-id`* `|` *`vlan-name`*`)`	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Not supported

Platform Support for Match Conditions for Non-IP Traffic

You can define port, VLAN, and router firewall filters for ingress and egress non-IP traffic on all EX Series switches. Table 83 on page 1607 summarizes support for match conditions on different bind points for ingress and egress non-IP traffic on different switches.

Table 83: Firewall Filter Match Condition Supported for Non-IP Traffic on Switches

Match Condition	Switch	Supported Bind Points	
		Ingress	Egress
`l2-encap-type llc-non-snap`	EX2200	Ports and VLANs	Ports and VLANs
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports and VLANs	Ports and VLANs
	EX3300	Ports and VLANs	Ports and VLANs
	EX4300	Not supported	Not supported
	EX4500	Ports and VLANs	Ports and VLANs
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs	Ports and VLANs

Platform Support for Actions for IPv4 Traffic

Table 84 on page 1607 summarizes the support for actions on different bind points for ingress and egress IPv4 traffic on different switches.

Table 84: Firewall Filter Actions Supported for IPv4 Traffic on Switches

Action	Switch	Supported Bind Points	
		Ingress	Egress
`accept`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 84: Firewall Filter Actions Supported for IPv4 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`discard`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 84: Firewall Filter Actions Supported for IPv4 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`reject` *`message-type`*	EX2200	Layer 3 interfaces	Not supported
	EX2300 and EX3400	Layer 3 interfaces	Not supported
	EX3200 and EX4200	Layer 3 interfaces	Not supported
	EX3300	Layer 3 interfaces	Not supported
	EX4300	Layer 3 interfaces	Not supported
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Not supported
	EX8200	Layer 3 interfaces	Not supported

Table 84: Firewall Filter Actions Supported for IPv4 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
`routing-instance` *routing-instance-name*	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported (EX2300) Layer 3 interfaces (EX3400)	Not supported
	EX3200 and EX4200	Layer 3 interfaces	Not supported
	EX3300	Layer 3 interfaces	Not supported
	EX4300	Layer 3 interfaces	Not supported
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Not supported
	EX8200	Layer 3 interfaces	Not supported
`vlan` *vlan-name*	EX2200	Ports and VLANs	Not supported
	EX2300 and EX3400	Ports and VLANs	Not supported
	EX3200 and EX4200	Ports and VLANs	Not supported
	EX3300	Ports and VLANs	Not supported
	EX4300	Ports and VLANs	Not supported
	EX4500	Ports and VLANs	Ports

Table 84: Firewall Filter Actions Supported for IPv4 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX6200	Ports and VLANs	Ports and VLANs
	EX8200	Ports and VLANs **NOTE**: Supported only when used in conjunction with the `interface` action modifier. On EX8200 Virtual Chassis, the `vlan` action is supported only for VLANs.	Not supported

Platform Support for Actions for IPv6 Traffic

Table 85 on page 1611 summarizes the support for actions on different bind points for ingress and egress IPv6 traffic.

Table 85: Firewall Filter Actions Supported for IPv6 Traffic on Switches

Action	Switch	Supported Bind Points	
		Ingress	Egress
`accept`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 85: Firewall Filter Actions Supported for IPv6 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`discard`	EX2200	Ports and VLANs, and Layer 3 interfaces	Ports and VLANs, and Layer 3 interfaces
	EX2300 and EX3400	Ports and VLANs, and Layer 3 interfaces	Ports and VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces

Table 85: Firewall Filter Actions Supported for IPv6 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
`reject` *`message-type`*	EX2200	Layer 3 interfaces	Not supported
	EX2300 and EX3400	Layer 3 interfaces	Not supported
	EX3200 and EX4200	Layer 3 interfaces	Not supported
	EX3300	Layer 3 interfaces	Not supported
	EX4300	Layer 3 interfaces	Not supported
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Not supported
	EX8200	Layer 3 interfaces	Not supported
`routing-instance` *`routing-instance-name`*	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported (EX2300) Layer 3 interfaces (EX3400)	Not supported

Table 85: Firewall Filter Actions Supported for IPv6 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Layer 3 interfaces	Not supported
	EX3300	Layer 3 interfaces	Not supported
	EX4300	Not supported	Not supported
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Layer 3 interfaces	Not supported
	EX8200	Layer 3 interfaces	Not supported
`vlan` *`vlan-name`*	EX2200	Ports and VLANs	Not supported
	EX2300 and EX3400	Ports and VLANs	Not supported
	EX3200 and EX4200	Ports and VLANs	Not supported
	EX3300	Ports and VLANs	Not supported
	EX4300	Ports and VLANs	Not supported
	EX4500	Ports and VLANs	Not supported
	EX6200	Ports and VLANs	Not supported

Table 85: Firewall Filter Actions Supported for IPv6 Traffic on Switches *(Continued)*

Action	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Ports and VLANs **NOTE**: Supported only when used in conjunction with the `interface` action modifier. On EX8200 Virtual Chassis, the `vlan` action is supported only for VLANs.	Not supported

Platform Support for Action Modifiers for IPv4 Traffic

Table 86 on page 1615 summarizes support for action modifiers on different bind points for ingress and egress IPv4 traffic on different switches.

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
`analyzer`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Not supported	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports, VLANs, and Layer 3 interfaces	Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`dscp`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Not supported	Not supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported
	EX8200	Layer 3 interfaces	Not supported
`count`	EX2200	VLANs and Layer 3 interfaces (`me0` interfaces only)	Layer 3 interfaces (`me0` interfaces only)

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 Interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	VLANs and Layer 3 interfaces (`me0` and `vme0` interfaces only)	Layer 3 interfaces (`me0` and `vme0` interfaces only)
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`forwarding-class` *class*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
`interface` *`interface-name`*	EX2200	Ports and VLANs	Not supported
	EX2300 and EX3400	Ports and VLANs	Ports and VLANs
	EX3200 and EX4200	Ports and VLANs	Not supported
	EX3300	Ports and VLANs	Not supported
	EX4300	Ports and VLANs	Not supported
	EX4500	Ports and VLANs	Not supported
	EX6200	Ports and VLANs	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Ports and VLANs **NOTE**: On EX8200 Virtual Chassis, the `interface` action modifier is supported only for VLANs.	Not supported
`log`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
`loss-priority (high | low)`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
`policer` *`policer-name`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`port-mirror`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported
	EX8200	Not supported	Not supported
`port-mirror-instance instance-name`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported
	EX8200	Not supported	Not supported
`syslog`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`three-color-policer`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interface	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 86: Firewall Filter Action Modifiers Supported for IPv4 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Not supported	Not supported

Platform Support for Action Modifiers for IPv6 Traffic

Table 87 on page 1624 summarizes support for action modifiers on different bind points for ingress and egress IPv6 traffic.

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
`analyzer`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Not supported	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and layer 3 interfaces	Not supported
	EX4500	Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
dscp	EX2200	Not supported	Not supported
	EX2300 and EX3400	Not supported	Not supported
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Not supported	Not supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Layer 3 interfaces	Not supported
`count`	EX2200	VLANs and Layer 3 interfaces (me0 and vme0 interfaces only)	Layer 3 interfaces (me0 and vme0 interfaces only)
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX3300	Layer 3 interfaces (`me0` and `vme0` interfaces only)	Layer 3 interfaces (`me0` and `vme0` interfaces only)
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`forwarding-class` *`class`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX3300	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
`interface` *`interface-name`*	EX2200	Ports and VLANs	Not supported
	EX2300 and EX3400	Ports and VLANs	Not supported
	EX3200 and EX4200	Ports and VLANs	Not supported
	EX3300	Ports and VLANs	Not supported
	EX4300	Ports and VLANs	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX4500	Ports and VLANs	Not supported
	EX6200	Ports and VLANs	Not supported
	EX8200	Ports and VLANs **NOTE**: On EX8200 Virtual Chassis, the `interface` action modifier is supported only for VLANs.	Not supported
`log`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`loss-priority (high | low)`	EX2200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX3300	Ports, VLANS, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
	EX8200	Ports, VLANs, and Layer 3 interfaces	Ports and Layer 3 interfaces
`policer` *`policer-name`*	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Ports, VLANs, and Layer 3 interfaces
	EX4500	Layer 3 interfaces	Layer 3 interfaces
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`port-mirror`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX6200	Not supported	Not supported
	EX8200	Not supported	Not supported
`port-mirror-instance instance-name`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported (EX2300) Ports, VLANs, and Layer 3 interfaces (EX3400)
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Not supported	Not supported
	EX8200	Not supported	Not supported
`syslog`	EX2200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Ports, VLANs, and Layer 3 interfaces	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX3300	Ports, VLAN, and Layer 3 interfaces	Not supported
	EX4300	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX4500	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX6200	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX8200	Ports, VLANs, and Layer 3 interfaces	Not supported
`three-color-policer`	EX2200	Not supported	Not supported
	EX2300 and EX3400	Ports, VLANs, and Layer 3 interfaces	Not supported
	EX3200 and EX4200	Not supported	Not supported
	EX3300	Not supported	Not supported
	EX4300	Not Supported	Not Supported
	EX4500	Not supported	Not supported
	EX6200	Not supported	Not supported

Table 87: Firewall Filter Action Modifiers Supported for IPv6 Traffic on Switches *(Continued)*

Action Modifier	Switch	Supported Bind Points	
		Ingress	Egress
	EX8200	Not Supported	Not Supported

RELATED DOCUMENTATION

- Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**
- Support for Match Conditions and Actions for Loopback Firewall Filters on Switches | **1633**
- Understanding Firewall Filter Match Conditions | **1543**
- Firewall Filter Configuration Statements Supported by Junos OS for EX Series Switches | **2254**
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**
- Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**

Support for Match Conditions and Actions for Loopback Firewall Filters on Switches

On EX Series Ethernet switches, a loopback interface is a gateway for all the control traffic that enters the Routing Engine of the switch. If you want to monitor this control traffic, you must configure a firewall filter on the loopback interface (lo0). Loopback firewall filters are applied only to packets that are sent to the Routing Engine CPU for further processing. Therefore, you can apply a firewall filter only in the ingress direction on the loopback interface.

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the values or fields that a packet must contain. You can define multiple, single, or no match conditions. If no match conditions are specified for the term, all packets are matched by default. The string that defines a match condition is called a *match statement*. The action is the action that the switch takes if a packet matches the match conditions for the specific term. Action modifiers are optional and specify one or more actions that the switch takes if a packet matches the match conditions for the specific term.

The following tables list match conditions, actions, and action modifiers that are supported for a firewall filter configured on a loopback interface on a switch:

- Table 88 on page 1634
- Table 89 on page 1636
- Table 90 on page 1637

For information on match conditions, actions, and action modifiers supported for a firewall filter configured on a network interface, see "Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches" on page 1570.

Table 88: Match Conditions for Firewall Filters on Loopback Interfaces for IPv4 and IPv6 Traffic—Support per Switch

Match Condition	EX2200	EX3200, EX4200	EX3300	EX4500	EX6200	EX8200
Match conditions for IPv4 traffic:						
destination-address	✓	✓	✓	✓	✓	✓
destination-port	✓	✓	✓	✓	✓	✓
destination-prefix-list	✓	✓	✓	✓	✓	✓
dscp	✓	✓	✓	✓	✓	✓
icmp-code	✓	✓	✓	✓	✓	✓
icmp-type	✓	✓	✓	✓	✓	✓
interface	✓	✓	✓	✓	✓	✓
is-fragment	✓	✓	✓	✓	–	–
packet-length	–	–	–	–	–	✓
precedence	✓	✓	✓	✓	✓	✓

Table 88: Match Conditions for Firewall Filters on Loopback Interfaces for IPv4 and IPv6 Traffic—Support per Switch *(Continued)*

Match Condition	EX2200	EX3200, EX4200	EX3300	EX4500	EX6200	EX8200
protocol	✓	✓	✓	✓	✓	✓
source-address	✓	✓	✓	✓	✓	✓
source-port	✓	✓	✓	✓	✓	✓
source-prefix-list	✓	✓	✓	✓	✓	✓
Match conditions for IPv6 traffic:						
ip6-destination-address	✓	✓	✓	✓	✓	✓
destination-port	✓	✓	✓	✓	✓	✓
destination-prefix-list	✓	✓	✓	✓	✓	✓
icmp-code	✓	✓	✓	✓	✓	✓
icmp-type	✓	✓	✓	✓	✓	✓
interface	✓	✓	✓	✓	✓	✓
next-header	✓	✓	✓	✓	✓	✓
packet-length	–	–	–	–	–	✓
source-address	✓	✓	✓	✓	✓	✓

Table 88: Match Conditions for Firewall Filters on Loopback Interfaces for IPv4 and IPv6 Traffic—Support per Switch *(Continued)*

Match Condition	EX2200	EX3200, EX4200	EX3300	EX4500	EX6200	EX8200
source-port	✓	✓	✓	✓	✓	✓
source-prefix-list	✓	✓	✓	✓	✓	✓
tcp-established	✓	✓	✓	✓	✓	–
tcp-flags	✓	✓	✓	✓	✓	–
tcp-initial	✓	✓	✓	✓	✓	–
traffic-class	✓	✓	✓	✓	✓	✓

Table 89: Actions for Firewall Filters on Loopback Interfaces for IPv4 and IPv6 Traffic—Support per Switch

Action	EX2200	EX3200, EX4200	EX3300	EX4500	EX6200	EX8200
Actions for IPv4 traffic:						
accept	✓	✓	✓	✓	✓	✓
discard	✓	✓	✓	✓	✓	✓
Actions for IPv6 traffic:						
accept	✓	✓	✓	✓	✓	✓
discard	✓	✓	✓	✓	✓	✓

Table 90: Action Modifiers for Firewall Filters on Loopback Interfaces for IPv4 and IPv6 Traffic—Support per Switch

Action	EX2200	EX3200, EX4200	EX3300	EX4500	EX6200	EX8200
Action modifiers for IPv4 traffic:						
count	–	✓	–	✓	✓	–
forwarding-class	✓	✓	✓	✓	–	✓
loss-priority	✓	✓	✓	✓	–	✓
Action modifiers for IPv6 traffic:						
count	–	✓	–	✓	–	–
forwarding-class	✓	✓	✓	✓	–	✓
loss-priority	✓	✓	✓	✓	–	✓

NOTE: On EX8200 switches, if an implicit or explicit `discard` action is configured on a loopback interface for IPv4 traffic, next hop resolve packets are accepted and allowed to pass through the switch. However, for IPv6 traffic, you must explicitly configure a rule to allow the neighbor discovery IPv6 resolve packets to pass through the switch.

RELATED DOCUMENTATION

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**

Platform Support for Firewall Filter Match Conditions, Actions, and Action Modifiers on EX Series Switches | **1570**

Understanding Firewall Filter Match Conditions | **1543**

Understanding How Firewall Filters Are Evaluated | 1550
Understanding How Firewall Filters Test a Packet's Protocol | 1649
Understanding the Use of Policers in Firewall Filters | 2210

Configuring Firewall Filters (CLI Procedure)

IN THIS SECTION

- Configuring a Firewall Filter | 1638
- Configuring a Term Specifically for IPv4 or IPv6 Traffic | 1643
- Applying a Firewall Filter to a Port on a Switch | 1644
- Applying a Firewall Filter to a Management Interface on a Switch | 1645
- Applying a Firewall Filter to a VLAN on a Network | 1646
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | 1648

You configure firewall filters on EX Series switches to control traffic that enters ports on the switch or enters and exits VLANs on the network and Layer 3 (routed) interfaces. To configure a firewall filter you must configure the filter and then apply it to a port, VLAN, or Layer 3 interface.

Configuring a Firewall Filter

Before you can apply a firewall filter to a port, VLAN, or Layer 3 interface, you must configure a firewall filter with the required details, such as type of family for the firewall filter, firewall filter name, and match conditions. A match condition in the firewall filter configuration can contain multiple terms that define the criteria for the match condition. For each term, you must specify an action to be performed if a packet matches the conditions in the term. For information on different match conditions and actions, see "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554.

To configure a firewall filter:

1. Configure the family address type for the firewall filter:

- For a firewall filter that is applied to a port or VLAN, specify the family address type **ethernet-switching** to filter Layer 2 (Ethernet) packets and Layer 3 (IP) packets, for example:

```
[edit firewall]
user@switch# set family ethernet-switching
```

- For a firewall filter that is applied to a Layer 3 (routed) interface:
  - To filter IPv4 packets, specify the family address type **inet**, for example:

```
[edit firewall]
user@switch# set family inet
```

  - To filter IPv6 packets, specify the family address type **inet6**, for example:

```
[edit firewall]
user@switch# set family inet6
```

NOTE: You can configure firewall filters for both IPv4 and IPv6 traffic on the same Layer 3 interface.

2. Specify the filter name:

```
[edit firewall family ethernet-switching]
user@switch# set filter ingress-port-filter
```

The filter name can contain letters, numbers, and hyphens (-) and can have a maximum of 64 characters. Each filter name must be unique.

3. If you want to apply a firewall filter to multiple interfaces and name individual firewall counters specific to each interface, configure the **interface-specific** option:

```
[edit firewall family ethernet-switching filter ingress-port-filter]
user@switch# set interface-specific
```

4. Specify a term name:

```
[edit firewall family ethernet-switching filter ingress-port-filter]
user@switch# set term term-one
```

The term name can contain letters, numbers, and hyphens (-) and can have a maximum of 64 characters.

A firewall filter can contain one or more terms. Each term name must be unique within a filter.

NOTE: The maximum number of terms allowed per firewall filter for EX Series switches is:

- 512 for EX2200 switches
- 1,436 for EX3300 switches

  NOTE: On EX3300 switches, if you add and delete filters with a large number of terms (on the order of 1000 or more) in the same commit operation, not all the filters are installed. You must add filters in one commit operation, and delete filters in a separate commit operation.

- 7,168 for EX3200 and EX4200 switches
- On EX4300 switches, following are the number of terms supported for ingress and egress traffic, for firewall filers configured on a port, VLAN and Layer 3 interface:
  - For ingress traffic:
    - 3,500 terms for firewall filters configured on a port
    - 3,500 terms for firewall filters configured on a VLAN
    - 7,000 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
    - 3,500 terms for firewall filers configured on Layer 3 interfaces for IPv6 traffic
  - For egress traffic:
    - 512 terms for firewall filters configured on a port

- 256 terms for firewall filters configured on a VLAN
- 512 terms for firewall filters configured on Layer 3 interfaces for IPv4 traffic
- 512 terms for firewall filers configured on Layer 3 interfaces for IPv6 traffic

NOTE: You can configure these maximum number of terms only when you configure one type of firewall filter (Port, VLAN, or Router (Layer 3) firewall filter) on the switch, and when storm control is not enabled on all interfaces in the switch.

- 1,200 for EX4500 and EX4550 switches
- 1,400 for EX6200 switches
- 32,768 for EX8200 switches

If you attempt to configure a firewall filter that exceeds these limits, the switch returns an error message when you commit the configuration.

5. For each firewall filter term, specify the match condition(s) that you want to include. The example below shows how to match packets from a given IP address and port:

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one]
user@switch# set from source-address 192.0.2.0
user@switch# set from source-port 80
```

You can specify one or more match conditions in a single `from` statement. For a match to occur, the packet must match all the conditions in the term.

The `from` statement is optional, but if included in a term, the `from` statement cannot be empty. If you omit the `from` statement, all packets are considered to match.

6. For each firewall filter term, specify the action to take if the packet matches all the conditions in that term.

You can specify an action and/or action modifiers:

- To specify a filter action, for example, to discard packets that match the conditions of the filter term:

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one]
user@switch#  set then discard
```

You can specify no more than one action per filter term.

- To specify an action modifier, for example, to count and classify packets in a forwarding class:

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one]
user@switch# set then count counter-one
user@switch# set then forwarding-class expedited-forwarding
```

In a `then` statement, you can specify the following action modifiers:

- **analyzer** *analyzer-name*—Mirror port traffic to a specified destination port or VLAN that is connected to a protocol analyzer application. An **analyzer** must be configured under the **ethernet-switching** family address type. See *Configuring Port Mirroring to Analyze Traffic (CLI Procedure)*.
- **count** *counter-name*—Count the number of packets that pass this filter term.

  NOTE: We recommend that you configure a counter for each term in a firewall filter, so that you can monitor the number of packets that match the conditions specified in each filter term.

- **forwarding-class** *class*—Classify packets in a forwarding class.
- **loss-priority** *priority*—Set the priority for dropping a packet.
- **policer** *policer-name*—Apply rate limiting to the traffic.
- **interface** *interface-name*—Forward the traffic to the specified interface, bypassing the switching lookup.
- **log**—Log the packet's header information in the Routing Engine.

If you omit the `then` statement or do not specify an action, packets that match all the conditions in the `from` statement are accepted. However, you must always explicitly configure an action and/or action modifier in the `then` statement. You can include no more than one action, but you can use any combination of action modifiers. For an action or action modifier to take effect, all conditions in the `from` statement must match.

NOTE: Implicit discard is also applicable to a firewall filter applied to the loopback interface, **lo0**.

On Juniper Networks EX8200 Ethernet Switches, if an implicit or explicit **discard** action is configured on a loopback interface for IPv4 traffic, next hop resolve packets are accepted and allowed to pass through the switch. However, for IPv6 traffic, you must explicitly configure a rule to allow the next hop IPv6 resolve packets to pass through the switch.

Configuring a Term Specifically for IPv4 or IPv6 Traffic

To configure a term in a firewall filter configuration specifically for IPv4 traffic:

1. Verify that neither `ether-type ipv6` nor `ip-version ipv6` is specified in the term in the configuration. By default, a configuration that does not contain either `ether-type ipv6` or `ip-version ipv6` in a term applies to IPv4 traffic.
2. (Optional) Perform one of these tasks:
   - Define `ether-type ipv4` in a term in the configuration.
   - Define `ip-version ipv4` in a term in the configuration.
   - Define both `ether-type ipv4` and `ip-version ipv4` in a term in the configuration.
   - Verify that neither `ether-type ipv6` nor `ip-version ipv6` is specified in a term in the configuration—by default, a configuration that does not contain either `ether-type ipv6` or `ip-version ipv6` in a term applies to IPv4 traffic if it does not contain `ether-type ipv6` or `ip-version ipv6`.
3. Ensure that other match conditions in the term are valid for IPv4 traffic.

To configure a term in a firewall filter configuration specifically for IPv6 traffic:

1. Perform one of these tasks:
   - Define `ether-type ipv6` in a term in the configuration.
   - Define `ip-version ipv6` in a term in the configuration.
   - Define both `ether-type ipv6` and `ip-version ipv4` in a term in the configuration.

   NOTE: By default, a configuration that does not contain either `ether-type ipv6` or `ip-version ipv6` in a term applies to IPv4 traffic.
2. Ensure that other match conditions in the term are valid for IPv6 traffic.

NOTE: If the term contains either of the match conditions `ether-type ipv6` or `ip-version ipv6`, with no other IPv6 match condition specified, all IPv6 traffic is matched.

NOTE: To configure a firewall filter for both IPv4 and IPv6 traffic, you must include two separate terms, one for IPv4 traffic and the other for IPv6 traffic.

Applying a Firewall Filter to a Port on a Switch

You can apply a firewall filter to a port on a switch to filter ingress or egress traffic on the switch. When you configure the firewall filter, you can specify any match condition, action, and action modifiers specified in "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554. The action specified in the match condition indicates the action for the matched packets in the ingress or egress traffic.

To apply a firewall filter to a port to filter ingress or egress traffic:

NOTE: For applying a firewall filter to a management interface, see "Applying a Firewall Filter to a Management Interface on a Switch" on page 1645

1. Specify the interface name and provide a meaningful description of the firewall filter and the interface to which the filter is applied:

```
[edit interfaces]
user@switch# set ge-0/0/1 description "filter to limit tcp traffic filter at trunk port for
employee-vlan and voice-vlan applied on the interface"
```

NOTE: Providing the description is optional.

2. Specify the unit number and family address type for the interface:

```
[edit interfaces]
user@switch# set ge-0/0/1 unit 0 family ethernet-switching
```

For firewall filters that are applied to ports, the family address type must be **ethernet-switching**.

3. To apply a firewall filter to filter packets that are entering a port:

```
[edit interfaces]
user@switch# set ge-0/0/1 unit 0 family ethernet-switching filter input ingress-port-filter
```

To apply a firewall filter to filter packets that are exiting a port:

```
[edit interfaces]
user@switch# set ge-0/0/1 unit 0 family ethernet-switching filter output egress-port-filter
```

NOTE: You can apply no more than one firewall filter per port, per direction.

Applying a Firewall Filter to a Management Interface on a Switch

You can configure and apply a firewall filter to a management interface to control traffic that is entering or exiting the interface on a switch. You can use utilities such as SSH or Telnet to connect to the management interface over the network and then use management protocols such as SNMP to gather statistical data from the switch. Similar to configuring a firewall filter on other types of interfaces, you can configure a firewall filter on a management interface using any match condition, action, and action modifier specified in "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554 except for the following action modifiers:

- loss-priority
- forwarding-class

You can apply a firewall filter to the management Ethernet interface on any EX Series switch. You can also apply a firewall filter to the virtual management Ethernet (VME) interface on the EX4200 switch. For more information on the management Ethernet interface and the VME interface, see *Interfaces Overview for Switches*.

To apply a firewall filter on the management interface to filter ingress or egress traffic:

1. Specify the interface name and provide a meaningful description of the firewall filter and the interface to which the filter is applied:

```
[edit interfaces]
user@switch# set me0 description "filter to limit tcp traffic filter at management interface"
```

NOTE: Providing the description is optional.

2. Specify the unit number and family address type for the management interface:

```
[edit interfaces]
user@switch# set me0 unit 0 family inet
```

NOTE: For firewall filters that are applied to management interfaces, the family address type can be either **inet** or **inet6**.

3. To apply a firewall filter to filter packets that are entering a management interface:

```
[edit interfaces]
user@switch# set me0 unit 0 family inet filter input ingress-port-filter
```

To apply a firewall filter to filter packets that are exiting a management interface:

```
[edit interfaces]
user@switch# set me0 unit 0 family inet filter output egress-port-filter
```

NOTE: You can apply no more than one firewall filter per management interface, per direction.

Applying a Firewall Filter to a VLAN on a Network

You can apply a firewall filter to a VLAN on a network to filter ingress or egress traffic on the network. To apply a firewall filter to a VLAN, specify the VLAN name and ID, and then apply the firewall filter to the VLAN. When you configure the firewall filter, you can specify any match condition, action, and action modifiers specified in "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554. The action specified in the match condition indicates the action for the matched packets in the ingress or egress traffic.

To apply a firewall filter to a VLAN:

1. Specify the VLAN name and VLAN ID and provide a meaningful description of the firewall filter and the VLAN to which the filter is applied:

```
[edit vlans]
user@switch# set employee-vlan vlan-id 20 vlan-description "filter to rate limit traffic
applied on employee-vlan"
```

NOTE: Providing the description is optional.

2. Apply firewall filters to filter packets that are entering or exiting the VLAN:
   - To apply a firewall filter to filter packets that are entering the VLAN:

```
[edit vlans]
user@switch#  set employee-vlan vlan-id 20 filter input ingress-vlan-filter
```

(On EX4300 switches) To apply a firewall filter to filter packets that are entering the VLAN:

```
[edit vlans]
user@switch#  set employee-vlan vlan-id 20 forwarding-options input ingress-vlan-filter
```

   - To apply a firewall filter to filter packets that are exiting the VLAN:

```
[edit vlans]
user@switch#  set employee-vlan vlan-id 20 filter output egress-vlan-filter
```

(On EX4300 switches) To apply a firewall filter to filter packets that are exiting the VLAN:

```
[edit vlans]
user@switch#  set employee-vlan vlan-id 20 forwarding-options output egress-vlan-filter
```

NOTE: You can apply no more than one firewall filter per VLAN, per direction.

Applying a Firewall Filter to a Layer 3 (Routed) Interface

You can apply a firewall filter to a Layer 3 (routed) interface to filter ingress or egress traffic on the switch. When you configure the firewall filter, you can specify any match condition, action, and action modifiers specified in "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554. The action specified in the match condition indicates the action for the matched packets in the ingress or egress traffic.

To apply a firewall filter to a Layer 3 interface on a switch:

1. Specify the interface name and provide a meaningful description of the firewall filter and the interface to which the filter is applied:

```
[edit interfaces]
user@switch# set ge-0/1/0 description "filter to count and monitor employee-vlan traffic
applied on layer 3 interface"
```

NOTE: Providing the description is optional.

2. Specify the unit number, family address type, and address for the interface:

```
[edit interfaces]
user@switch# set ge-0/1/0 unit 0 family inet address 10.10.10.1/24
```

For firewall filters applied to Layer 3 interfaces, the family address type must be **inet** (for IPv4 traffic) or **inet6** (for IPv6 traffic).

3. You can apply firewall filters to filter packets that are entering or exiting a Layer 3 (routed) interface:
   - To apply a firewall filter to filter packets that are entering a Layer 3 interface:

```
[edit interfaces]
user@switch# set ge-0/1/0 unit 0 family inet address 10.10.10.1/24 filter input ingress-
router-filter
```

   - To apply a firewall filter to filter packets that are exiting a Layer 3 interface:

```
[edit interfaces]
user@switch# set ge-0/1/0 unit 0 family inet address 10.10.10.1/24 filter output egress-
router-filter
```

NOTE: When you apply a filter to an IRB interface associated with a given VLAN, the filter is executed on any Layer 3 interface with a matching VLAN ID. This is because the filter matches on all Layer 3 interfaces with the corresponding VLAN tag.

NOTE: You can apply no more than one firewall filter per Layer 3 interface, per direction.

RELATED DOCUMENTATION

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**

Example: Configuring a Firewall Filter on a Management Interface on an EX Series Switch | **1680**

Verifying That Firewall Filters Are Operational | **1866**

Monitoring Firewall Filter Traffic | **1867**

Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**

Understanding How Firewall Filters Test a Packet's Protocol

When examining match conditions, Juniper Networks Junos operating system (Junos OS) for Juniper Networks EX Series Ethernet Switches tests only the field that is specified. The software does not implicitly test the IP header to determine whether a packet is an IP packet. Therefore, in some cases, you must specify **protocol** field match conditions in conjunction with other match conditions to ensure that the filters are performing the expected matches.

If you specify a protocol match condition or a match of the ICMP type or TCP flags field, there is no implied protocol match. For the following match conditions, you must explicitly specify the protocol match condition in the same term:

- **destination-port**—Specify the match **protocol tcp** or **protocol udp**.
- **source-port**—Specify the match **protocol tcp** or **protocol udp**.

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

RELATED DOCUMENTATION

Firewall Filters for EX Series Switches Overview | 1535

Understanding Firewall Filter Match Conditions | 1543

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Understanding Filter-Based Forwarding for EX Series Switches

Administrators of Juniper Networks EX Series Ethernet Switches can use firewall filters in conjunction with virtual routing instances to specify different routes for packets to travel in their networks. To set up this feature, which is called filter-based forwarding, you specify a filter and match criteria and then specify the virtual routing instance to send packets to.

You might want to use filter-based forwarding to route specific types of traffic through a firewall or security device before the traffic continues on its path. You can also use filter-based forwarding to give certain types of traffic preferential treatment or to improve load balancing of switch traffic.

RELATED DOCUMENTATION

Understanding Virtual Routing Instances on EX Series Switches

Firewall Filters for EX Series Switches Overview | 1535

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | 1685

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches

IN THIS SECTION

- Requirements | 1651
- Overview | 1651
- Configuring an Ingress Port Firewall Filter to Prioritize Voice Traffic and Rate-Limit TCP and ICMP Traffic | 1656
- Configuring a VLAN Ingress Firewall Filter to Prevent Rogue Devices from Disrupting VoIP Traffic | 1664

- Configuring a VLAN Firewall Filter to Count, Monitor, and Analyze Egress Traffic on the Employee VLAN | **1668**
- Configuring a VLAN Firewall Filter to Restrict Guest-to-Employee Traffic and Peer-to-Peer Applications on the Guest VLAN | **1671**
- Configuring a Router Firewall Filter to Give Priority to Egress Traffic Destined for the Corporate Subnet | **1674**
- Verification | **1676**

This example shows how to configure and apply firewall filters to control traffic that is entering or exiting a port on the switch, a VLAN on the network, and a Layer 3 interface on the switch. Firewall filters define the rules that determine whether to forward or deny packets at specific processing points in the packet flow.

Requirements

This example uses the following software and hardware components:

- Junos OS Release 9.0 or later for EX Series switches.
- Two Juniper Networks EX3200-48T switches: one to be used as an access switch, the other to be used as a distribution switch
- One Juniper Networks EX-UM-4SFP uplink module
- One Juniper Networks J-series router

Before you configure and apply the firewall filters in this example, be sure you have:

- An understanding of firewall filter concepts, policers, and CoS
- Installed the uplink module in the distribution switch. See Installing an Uplink Module in an EX3200 Switch.

Overview

IN THIS SECTION

- Network Topology | **1654**

This configuration example show how to configure and apply firewall filters to provide rules to evaluate the contents of packets and determine when to discard, forward, classify, count, and analyze packets that are destined for or originating from the EX Series switches that handle all `voice-vlan`, `employee-vlan`, and `guest-vlan` traffic. Table 91 on page 1652 shows the firewall filters that are configured for the EX Series switches in this example.

Table 91: Configuration Components: Firewall Filters

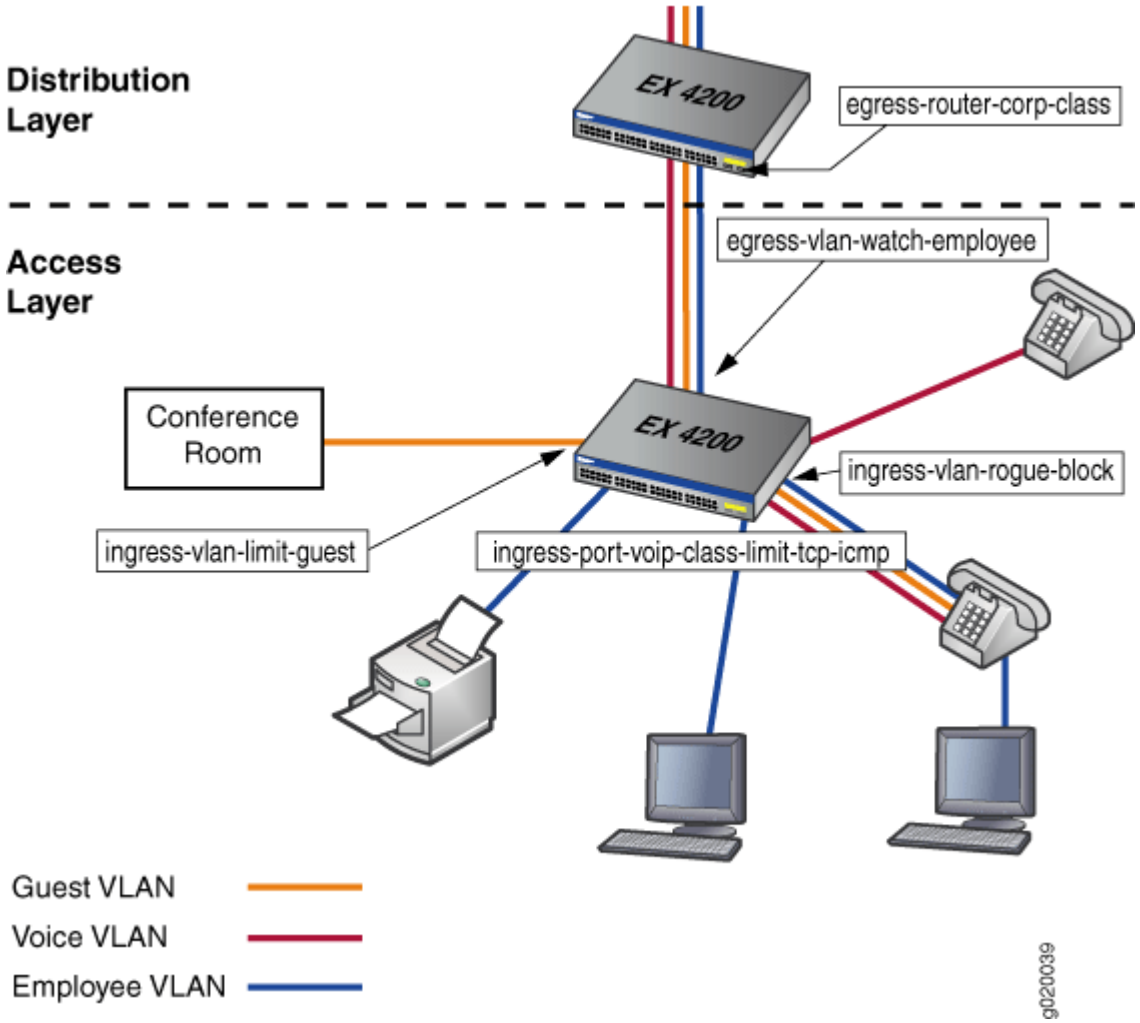
Component	Purpose/Description
Port firewall filter, `ingress-port-voip-class-limit-tcp-icmp`	This firewall filter performs two functions: • Assigns priority queueing to packets with a source MAC address that matches the phone MAC addresses. The forwarding class `expedited-forwarding` provides low loss, low delay, low jitter, assured bandwidth, and end-to-end service for all `voice-vlan` traffic. • Performs rate limiting on packets that enter the ports for `employee-vlan`. The traffic rate for TCP and ICMP packets is limited to 1 Mbps with a burst size up to 30,000 bytes. This firewall filter is applied to port interfaces on the access switch.
VLAN firewall filter, `ingress-vlan-rogue-block`	Prevents rogue devices from using HTTP sessions to mimic the gatekeeper device that manages call registration, admission, and call status for VoIP calls. Only TCP or UDP ports should be used; and only the gatekeeper uses HTTP. That is, all `voice-vlan` traffic on TCP ports should be destined for the gatekeeper device. This firewall filter applies to all phones on `voice-vlan`, including communication between any two phones on the VLAN and all communication between the gatekeeper device and VLAN phones. This firewall filter is applied to VLAN interfaces on the access switch.
VLAN firewall filter, `egress-vlan-watch-employee`	Accepts `employee-vlan` traffic destined for the corporate subnet, but does not monitor this traffic. Employee traffic destined for the Web is counted and analyzed. This firewall filter is applied to vlan interfaces on the access switch.
VLAN firewall filter, `ingress-vlan-limit-guest`	Prevents guests (non-employees) from talking with employees or employee hosts on `employee-vlan`. Also prevents guests from using peer-to-peer applications on `guest-vlan`, but allows guests to access the Web. This firewall filter is applied to VLAN interfaces on the access switch.

Table 91: Configuration Components: Firewall Filters *(Continued)*

Component	Purpose/Description
Router firewall filter, `egress-router-corp-class`	Prioritizes `employee-vlan` traffic, giving highest forwarding-class priority to employee traffic destined for the corporate subnet. This firewall filter is applied to a routed port (Layer 3 uplink module) on the distribution switch.

Figure 72 on page 1653 shows the application of port, VLAN, and Layer 3 routed firewall filters on the switch.

Figure 72: Application of Port, VLAN, and Layer 3 Routed Firewall Filters



Network Topology

The topology for this configuration example consists of one EX-3200-48T switch at the access layer, and one EX-3200-48T switch at the distribution layer. The distribution switch's uplink module is configured to support a Layer 3 connection to a J-series router.

The EX Series switches are configured to support VLAN membership. Table 92 on page 1654 shows the VLAN configuration components for the VLANs.

Table 92: Configuration Components: VLANs

VLAN Name	VLAN ID	VLAN Subnet and Available IP Addresses	VLAN Description
voice-vlan	10	192.0.2.0/28 192.0.2.1through 192.0.2.14 192.0.2.15 is the subnet's broadcast address	Voice VLAN used for employee VoIP traffic
employee-vlan	20	192.0.2.16/28 192.0.2.17 through 192.0.2.30 192.0.2.31 is the subnet's broadcast address	VLAN standalone PCs, PCs connected to the network through the hub in VoIP telephones, wireless access points, and printers. This VLAN completely includes the voice VLAN. Two VLANs (voice-vlan and employee-vlan) must be configured on the ports that connect to the telephones.

Table 92: Configuration Components: VLANs *(Continued)*

VLAN Name	VLAN ID	VLAN Subnet and Available IP Addresses	VLAN Description
guest-vlan	30	192.0.2.32/28 192.0.2.33 through 192.0.2.46 192.0.2.47 is the subnet's broadcast address	VLAN for guests' data devices (PCs). The scenario assumes that the corporation has an area open to visitors, either in the lobby or in a conference room, that has a hub to which visitors can plug in their PCs to connect to the Web and to their company's VPN.
camera-vlan	40	192.0.2.48/28 192.0.2.49 through 192.0.2.62 192.0.2.63 is the subnet's broadcast address	VLAN for the corporate security cameras.

Ports on the EX Series switches support Power over Ethernet (PoE) to provide both network connectivity and power for VoIP telephones connecting to the ports. Table 93 on page 1655 shows the switch ports that are assigned to the VLANs and the IP and MAC addresses for devices connected to the switch ports:

Table 93: Configuration Components: Switch Ports on a 48-Port All-PoE Switch

Switch and Port Number	VLAN Membership	IP and MAC Addresses	Port Devices
ge-0/0/0, ge-0/0/1	voice-vlan, employee-vlan	IP addresses: 192.0.2.1 through 192.0.2.2 MAC addresses: 00.00.5E.00.53.01, 00.00.5E.00.53.02	Two VoIP telephones, each connected to one PC.
ge-0/0/2, ge-0/0/3	employee-vlan	192.0.2.17 through 192.0.2.18	Printer, wireless access points

Table 93: Configuration Components: Switch Ports on a 48-Port All-PoE Switch *(Continued)*

Switch and Port Number	VLAN Membership	IP and MAC Addresses	Port Devices
ge-0/0/4, ge-0/0/5	guest-vlan	192.0.2.34 through 192.0.2.35	Two hubs into which visitors can plug in their PCs. Hubs are located in an area open to visitors, such as a lobby or conference room
ge-0/0/6, ge-0/0/7	camera-vlan	192.0.2.49 through 192.0.2.50	Two security cameras
ge-0/0/9	voice-vlan	IP address: 192.0.2.14 MAC address:00.05.5E.00.53.0E	Gatekeeper device. The gatekeeper manages call registration, admission, and call status for VoIP phones.
ge-0/1/0		IP address: 192.0.2.65	Layer 3 connection to a router; note that this is a port on the switch's uplink module

Configuring an Ingress Port Firewall Filter to Prioritize Voice Traffic and Rate-Limit TCP and ICMP Traffic

IN THIS SECTION

- Procedure | 1657

To configure and apply firewall filters for port, VLAN, and router interfaces, perform these tasks:

Procedure

CLI Quick Configuration

To quickly configure and apply a port firewall filter to prioritize voice traffic and rate-limit packets that are destined for the `employee-vlan` subnet, copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set firewall policer tcp-connection-policer if-exceeding burst-size-
limit 30k bandwidth-limit 1m
                        set firewall policer tcp-connection-policer then
discard
                        set firewall policer icmp-connection-policer if-exceeding burst-size-
limit 30k bandwidth-limit 1m
                        set firewall policer icmp-connection-policer then
discard
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term voip-high from source-mac-address 00.00.5E.00.53.01
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term voip-high from source-mac-address 00.00.5E.00.53.02
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term voip-high from protocol udp
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term voip-high then forwarding-class expedited-forwarding
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term voip-high then loss-priority low
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term network-control from precedence net-control
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term network-control then forwarding-class network-control
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term network-control then loss-priority low
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term tcp-connection from destination-address 192.0.2.16/28
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term tcp-connection from protocol tcp
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term tcp-connection then policer tcp-connection-policer
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term tcp-connection then count tcp-counter
                        set firewall family ethernet-switching filter ingress-port-voip-class-
```

```
limit-tcp-icmp term tcp-connection then forwarding-class best-effort
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term tcp-connection then loss-priority high
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection from destination-address 192.0.2.16/28
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection from protocol icmp
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection then policer icmp-connection-policer
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection then count icmp-counter
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection then forwarding-class best-effort
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term icmp-connection then loss-priority high
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term best-effort then forwarding-class best-effort
                        set firewall family ethernet-switching filter ingress-port-voip-class-
limit-tcp-icmp term best-effort then loss-priority high
                        set interfaces ge-0/0/0 description "voice priority and tcp and icmp
traffic rate-limiting filter at ingress port"
                        set interfaces ge-0/0/0 unit 0 family ethernet-switching filter input
ingress-port-voip-class-limit-tcp-icmp
                        set interfaces ge-0/0/1 description "voice priority and tcp and icmp
traffic rate-limiting filter at ingress port"
                        set interfaces ge-0/0/1 unit 0 family ethernet-switching filter input
ingress-port-voip-class-limit-tcp-icmp
                        set class-of-service schedulers voice-high buffer-size percent
15
                        set class-of-service schedulers voice-high  priority
high
                        set class-of-service schedulers net-control buffer-size percent
10
                        set class-of-service schedulers net-control priority
high
                        set class-of-service schedulers best-effort buffer-size percent
75
                        set class-of-service schedulers best-effort priority
low
                        set class-of-service scheduler-maps ethernet-diffsrv-cos-map forwarding-
class expedited-forwarding scheduler voice-high
                        set class-of-service scheduler-maps ethernet-diffsrv-cos-map forwarding-
class network-control scheduler net-control
```

```
                        set class-of-service scheduler-maps ethernet-diffsrv-cos-map forwarding-
class best-effort scheduler best-effort
```

Step-by-Step Procedure

To configure and apply a port firewall filter to prioritize voice traffic and rate-limit packets that are destined for the `employee-vlan` subnet:

1. Define the policers `tcp-connection-policer` and `icmp-connection-policer`:

```
[edit]
user@switch# set firewall policer tcp-connection-policer if-exceeding burst-size-limit 30k
bandwidth-limit 1m
user@switch# set firewall policer tcp-connection-policer then discard
user@switch# set firewall policer icmp-connection-policer if-exceeding burst-size-limit 30k
bandwidth-limit 1m
user@switch# set firewall policer icmp-connection-policer then
discard
```

2. Define the firewall filter `ingress-port-voip-class-limit-tcp-icmp`:

```
[edit firewall]
user@switch# set family ethernet-switching filter ingress-port-voip-class-limit-tcp-
icmp
```

3. Define the term `voip-high`:

```
[edit firewall family ethernet-switching filter ingress-port-voip-class-limit-tcp-icmp ]
user@switch# set term voip-high from source-mac-address 00.00.5E.00.53.01
user@switch# set term voip-high from source-mac-address 00.00.5E.00.53.02
user@switch# set term voip-high from protocol udp
user@switch# set term voip-high then forwarding-class expedited-forwarding
user@switch# set term voip-high then loss-priority low
```

4. Define the term `network-control`:

```
[edit firewall family ethernet-switching filter ingress-port-voip-class-limit-tcp-icmp ]
user@switch# set term network-control from precedence net-control
```

```
user@switch# set term network-control then forwarding-class network-control
user@switch# set term network-control then loss-priority low
```

5. Define the term `tcp-connection` to configure rate limits for TCP traffic:

```
[edit firewall family ethernet-switching filter ingress-port-voip-class-limit-tcp-icmp]
user@switch# set term tcp-connection from destination-address 192.0.2.16/28
user@switch# set term tcp-connection from protocol tcp
user@switch# set term tcp-connection then policer tcp-connection-policer
user@switch# set term tcp-connection then count tcp-counter
user@switch# set term tcp-connection then forwarding-class best-effort
user@switch# set term tcp-connection then loss-priority high
```

6. Define the term `icmp-connection` to configure rate limits for ICMP traffic:

```
[edit firewall family ethernet-switching filter ingress-port-voip-class-limit-tcp-icmp]
user@switch# set term icmp-connection from destination-address 192.0.2.16/28
user@switch# set term icmp-connection from protocol icmp
user@switch# set term icmp-connection then policer icmp-policer
user@switch# set term icmp-connection then count icmp-counter
user@switch# set term icmp-connection then forwarding-class best-effort
user@switch# set term icmp-connection then loss-priority high
```

7. Define the term `best-effort` with no match conditions for an implicit match on all packets that did not match any other term in the firewall filter:

```
[edit firewall family ethernet-switching filter ingress-port-voip-class-limit-tcp-icmp]
user@switch# set term best-effort then forwarding-class best-effort
user@switch# set term best-effort then loss-priority high
```

8. Apply the firewall filter `ingress-port-voip-class-limit-tcp-icmp` as an input filter to the port interfaces for `employee-vlan`:

```
[edit interfaces]
user@switch# set ge-0/0/0 description "voice priority and tcp and icmp traffic rate-
limiting filter at ingress port"
user@switch# set ge-0/0/0 unit 0 family ethernet-switching filter input ingress-port-voip-
class-limit-tcp-icmp
user@switch# set ge-0/0/1 description "voice priority and tcp and icmp traffic rate-
```

```
limiting filter at ingress port"
user@switch# set ge-0/0/1 unit 0 family ethernet-switching filter input ingress-port-voip-
class-limit-tcp-icmp
```

9. Configure the parameters that are desired for the different schedulers.

> **NOTE**: When you configure parameters for the schedulers, define the numbers to match your network traffic patterns.

```
[edit class-of-service]
user@switch# set schedulers voice-high buffer-size percent 15
user@switch# set schedulers voice-high priority  high
user@switch# set schedulers network-control buffer-size percent 10
user@switch# set schedulers network-control priority high
user@switch# set schedulers best-effort buffer-size percent 75
user@switch# set schedulers best-effort priority low
```

10. Assign the forwarding classes to schedulers with a scheduler map:

```
[edit class-of-service]
user@switch# set scheduler-maps ethernet-diffsrv-cos-map
user@switch# set scheduler-maps ethernet-diffsrv-cos-map forwarding-class expedited-
forwarding scheduler voice-high
user@switch# set scheduler-maps ethernet-diffsrv-cos-map forwarding-class network-control
scheduler net-control
user@switch# set scheduler-maps ethernet-diffsrv-cos-map forwarding-class best-effort
scheduler best-effort
```

11. Associate the scheduler map with the outgoing interface:

```
[edit class-of-service]
user@switch# set interfaces ge-0/1/0 scheduler-map ethernet-diffsrv-cos-
map
```

Results

Display the results of the configuration:

```
user@switch# show
    firewall {
        policer tcp-connection-policer {
            if-exceeding {
                bandwidth-limit 1m;
                burst-size-limit 30k;
            }
            then {
                discard;
            }
        }
        policer icmp-connection-policer {
            if-exceeding {
                bandwidth-limit 1m;
                burst-size-limit 30k;
            }
            then {
                discard;
            }
        }
        family ethernet-switching {
            filter ingress-port-voip-class-limit-tcp-icmp {
                 term voip-high {
                    from {
                        destination-mac-address 00.00.5E.00.53.01;
                            destination-mac-address 00.00.5E.00.53.02;
                        protocol udp;
                    }
                    then {
                        forwarding-class expedited-forwarding;
                        loss-priority low;
                    }
                }
                term network-control {
                    from {
                        precedence net-control ;
                    }
                    then {
```

```
                        forwarding-class network-control;
                        loss-priority low;
                    }
                }
                term tcp-connection {
                    from {
                        destination-address 192.0.2.16/28;
                        protocol tcp;
                    }
                    then {
                        policer tcp-connection-policer;
                            count tcp-counter;
                            forwarding-class best-effort;
                        loss-priority high;
                    }
                }
                term icmp-connection
                    from {
                        protocol icmp;
                    }
                    then {
                         policer icmp-connection-policer;
                            count icmp-counter;
                            forwarding-class best-effort;
                        loss-priority high;
                    }
                }
                term best-effort {
                    then {
                        forwarding-class best-effort;
                        loss-priority high;
                    }
                }
            }
        }
    }
    interfaces {
        ge-0/0/0 {
            description "voice priority and tcp and icmp traffic rate-limiting filter at ingress
port";
            unit 0 {
                family ethernet-switching {
                    filter {
```

```
                        input ingress-port-voip-class-limit-tcp-icmp;
                    }
                }
            }
        }
        ge-0/0/1 {
            description "voice priority and tcp and icmp traffic rate-limiting filter at ingress
port";
                unit 0 {
                family ethernet-switching {
                    filter {
                        input ingress-port-voip-class-limit-tcp-icmp;
                    }
                }
            }
        }
    }
    scheduler-maps {
        ethernet-diffsrv-cos-map {
            forwarding-class expedited-forwarding scheduler voice-high;
                forwarding-class network-control scheduler net-control;
            forwarding-class best-effort scheduler best-effort;
        }
    }
    interfaces {
        ge/0/1/0 {
            scheduler-map ethernet-diffsrv-cos-map;
        }
    }
```

Configuring a VLAN Ingress Firewall Filter to Prevent Rogue Devices from Disrupting VoIP Traffic

IN THIS SECTION

- Procedure | 1665

To configure and apply firewall filters for port, VLAN, and router interfaces, perform these tasks:

Procedure

CLI Quick Configuration

To quickly configure a VLAN firewall filter on `voice-vlan` to prevent rogue devices from using HTTP sessions to mimic the gatekeeper device that manages VoIP traffic, copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term to-gatekeeper from destination-address 192.0.2.14
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term to-gatekeeper from destination-port 80
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term to-gatekeeper then accept
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term from-gatekeeper from source-address 192.0.2.14
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term from-gatekeeper from source-port 80
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term from-gatekeeper then accept
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term not-gatekeeper from destination-port 80
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term not-gatekeeper then count rogue-counter
                        set firewall family ethernet-switching filter ingress-vlan-rogue-block
term not-gatekeeper then discard
                        set vlans voice-vlan description "block rogue devices on voice-
vlan"
                        set vlans voice-vlan filter input ingress-vlan-rogue-
block
```

Step-by-Step Procedure

To configure and apply a VLAN firewall filter on `voice-vlan` to prevent rogue devices from using HTTP to mimic the gatekeeper device that manages VoIP traffic:

1. Define the firewall filter `ingress-vlan-rogue-block` to specify filter matching on the traffic you want to permit and restrict:

```
[edit firewall]
user@switch# set family ethernet-switching filter ingress-vlan-rogue-
block
```

2. Define the term `to-gatekeeper` to accept packets that match the destination IP address of the gatekeeper:

```
[edit firewall family ethernet-switching filter ingress-vlan-rogue-block]
user@switch# set term to-gatekeeper from destination-address 192.0.2.14
user@switch# set term to-gatekeeper from destination-port 80
user@switch# set term to-gatekeeper then accept
```

3. Define the term `from-gatekeeper` to accept packets that match the source IP address of the gatekeeper:

```
[edit firewall family ethernet-switching filter ingress-vlan-rogue-block]
user@switch# set term from-gatekeeper from source-address 192.0.2.14
user@switch# set term from-gatekeeper from source-port 80
user@switch# set term from-gatekeeper then accept
```

4. Define the term `not-gatekeeper` to ensure all `voice-vlan` traffic on TCP ports is destined for the gatekeeper device:

```
[edit firewall family ethernet-switching filter ingress-vlan-rogue-block]
user@switch# set term not-gatekeeper from destination-port 80
user@switch# set term not-gatekeeper then count rogue-counter
user@switch# set term not-gatekeeper then discard
```

5. Apply the firewall filter `ingress-vlan-rogue-block` as an input filter to the VLAN interface for the VoIP telephones:

```
[edit]
user@switch# set vlans voice-vlan description "block rogue devices on voice-vlan"
user@switch# set vlans voice-vlan filter input ingress-vlan-rogue-block
```

Results

Display the results of the configuration:

```
user@switch# show
    firewall {
        family ethernet-switching {
            filter ingress-vlan-rogue-block {
                term to-gatekeeper {
                    from {
                        destination-address 192.0.2.14/32
                        destination-port 80;
                    }
                    then {
                        accept;
                    }
                }
                term from-gatekeeper {
                    from {
                        source-address 192.0.2.14/32
                        source-port 80;
                    }
                    then {
                        accept;
                    }
                }
                term not-gatekeeper {
                    from {
                        destination-port 80;
                    }
                    then {
                        count rogue-counter;
                        discard;
                    }
                }
        }
    vlans {
        voice-vlan {
            description "block rogue devices on voice-vlan";
            filter {
                 input ingress-vlan-rogue-block;
            }
```

```
        }
    }
```

Configuring a VLAN Firewall Filter to Count, Monitor, and Analyze Egress Traffic on the Employee VLAN

IN THIS SECTION

- Procedure | 1668

To configure and apply firewall filters for port, VLAN, and router interfaces, perform these tasks:

Procedure

CLI Quick Configuration

A firewall filter is configured and applied to VLAN interfaces to filter `employee-vlan` egress traffic. Employee traffic destined for the corporate subnet is accepted but not monitored. Employee traffic destined for the Web is counted and analyzed.

To quickly configure and apply a VLAN firewall filter, copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set firewall family ethernet-switching filter egress-vlan-watch-
employee term employee-to-corp from destination-address 192.0.2.16/28
                        set firewall family ethernet-switching filter egress-vlan-watch-
employee term employee-to-corp then accept
                        set firewall family ethernet-switching filter egress-vlan-watch-
employee term employee-to-web from destination-port 80
                        set firewall family ethernet-switching filter egress-vlan-watch-
employee term employee-to-web then count employee-web-counter
                        set firewall family ethernet-switching filter egress-vlan-watch-
employee term employee-to-web then analyzer employee-monitor
                        set vlans employee-vlan description "filter at egress VLAN to count and
analyze employee to Web traffic"
                        set vlans employee-vlan filter output egress-vlan-watch-
employee
```

Step-by-Step Procedure

To configure and apply an egress port firewall filter to count and analyze `employee-vlan` traffic that is destined for the Web:

1. Define the firewall filter `egress-vlan-watch-employee`:

```
[edit firewall]
user@switch# set family ethernet-switching filter egress-vlan-watch-
employee
```

2. Define the term `employee-to-corp` to accept but not monitor all `employee-vlan` traffic destined for the corporate subnet:

```
[edit firewall family ethernet-switching filter egress-vlan-watch-employee]
user@switch# set term employee-to-corp from destination-address 192.0.2.16/28
user@switch# set term employee-to-corp then accept
```

3. Define the term `employee-to-web` to count and monitor all `employee-vlan` traffic destined for the Web:

```
[edit firewall family ethernet-switching filter egress-vlan-watch-employee]
user@switch# set term employee-to-web from destination-port 80
user@switch# set term employee-to-web then count employee-web-counter
user@switch# set term employee-to-web then analyzer employee-monitor
```

NOTE: See Example: Configuring Port Mirroring for Local Monitoring of Employee Resource Use on EX Series Switches for information about configuring the `employee-monitor` analyzer.

4. Apply the firewall filter `egress-vlan-watch-employee` as an output filter to the port interfaces for the VoIP telephones:

```
[edit]
user@switch# set vlans employee-vlan description "filter at egress VLAN to count and analyze
employee to Web traffic"
user@switch# set vlans employee-vlan filter output egress-vlan-watch-
employee
```

Results

Display the results of the configuration:

```
user@switch# show
    firewall {
        family ethernet-switching {
            filter egress-vlan-watch-employee {
                term employee-to-corp {
                    from {
                        destination-address 192.0.2.16/28
                    }
                    then {
                        accept;
                    }
                }
                term employee-to-web {
                    from {
                        destination-port 80;
                    }
                    then {
                        count employee-web-counter:
                        analyzer employee-monitor;
                    }
                }
            }
        }
    }
    vlans {
        employee-vlan {
            description "filter at egress VLAN to count and analyze employee to Web traffic";
            filter {
                 output egress-vlan-watch-employee;
            }
        }
    }
```

Configuring a VLAN Firewall Filter to Restrict Guest-to-Employee Traffic and Peer-to-Peer Applications on the Guest VLAN

IN THIS SECTION

- Procedure | 1671

To configure and apply firewall filters for port, VLAN, and router interfaces, perform these tasks:

Procedure

CLI Quick Configuration

In the following example, the first filter term permits guests to talk with other guests but not employees on `employee-vlan`. The second filter term allows guests Web access but prevents them from using peer-to-peer applications on `guest-vlan`.

To quickly configure a VLAN firewall filter to restrict guest-to-employee traffic, blocking guests from talking with employees or employee hosts on `employee-vlan` or attempting to use peer-to-peer applications on `guest-vlan`, copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set firewall family ethernet-switching filter ingress-vlan-limit-guest
term guest-to-guest from destination-address 192.0.2.33/28
                        set firewall family ethernet-switching filter ingress-vlan-limit-guest
term guest-to-guest then accept
                        set firewall family ethernet-switching filter ingress-vlan-limit-guest
term no-guest-employee-no-peer-to-peer from destination-mac-address
00.05.5E.00.00.DF
                        set firewall family ethernet-switching filter ingress-vlan-limit-guest
term no-guest-employee-no-peer-to-peer then accept
                        set vlans guest-vlan description "restrict guest-to-employee traffic
and peer-to-peer applications on guest VLAN"
                        set vlans guest-vlan forwarding-options filter input ingress-vlan-limit-
guest
```

Step-by-Step Procedure

To configure and apply a VLAN firewall filter to restrict guest-to-employee traffic and peer-to-peer applications on guest-vlan:

1. Define the firewall filter ingress-vlan-limit-guest:

```
[edit firewall]
                              set firewall family ethernet-switching filter ingress-vlan-limit-
guest
```

2. Define the term guest-to-guest to permit guests on the guest-vlan to talk with other guests but not employees on the employee-vlan:

```
[edit firewall family ethernet-switching filter ingress-vlan-limit-guest]
user@switch# set term guest-to-guest from destination-address 192.0.2.33/28
user@switch# set term guest-to-guest then accept
```

3. Define the term no-guest-employee-no-peer-to-peer to allow guests on guest-vlan Web access but prevent them from using peer-to-peer applications on the guest-vlan.

> **NOTE**: The destination-mac-address is the default gateway, which for any host in a VLAN is the next-hop router.

```
[edit firewall family ethernet-switching filter ingress-vlan-limit-guest]
user@switch# set term no-guest-employee-no-peer-to-peer from destination-mac-address
00.05.5E.00.00.DF
user@switch# set term no-guest-employee-no-peer-to-peer then accept
```

4. Apply the firewall filter ingress-vlan-limit-guest as an input filter to the interface for guest-vlan :

```
[edit]
user@switch# set vlans guest-vlan description "restrict guest-to-employee traffic and peer-to-
peer applications on guest VLAN"
user@switch# set vlans guest-vlan forwarding-options filter input ingress-vlan-limit-
guest
```

Results

Display the results of the configuration:

```
user@switch# show
    firewall {
        family ethernet-switching {
            filter ingress-vlan-limit-guest {
                term guest-to-guest {
                    from {
                        destination-address 192.0.2.33/28;
                    }
                    then {
                        accept;
                    }
                }
                term no-guest-employee-no-peer-to-peer {
                    from {
                        destination-mac-address 00.05.5E.00.00.DF;
                    }
                    then {
                        accept;
                    }
                }
            }
        }
    }
    vlans {
        guest-vlan {
            description "restrict guest-to-employee traffic and peer-to-peer applications on
guest VLAN";
            filter {
                input ingress-vlan-limit-guest;
            }
        }
    }
```

Configuring a Router Firewall Filter to Give Priority to Egress Traffic Destined for the Corporate Subnet

IN THIS SECTION

- Procedure | 1674

To configure and apply firewall filters for port, VLAN, and router interfaces, perform these tasks:

Procedure

CLI Quick Configuration

To quickly configure a firewall filter for a routed port (Layer 3 uplink module) to filter `employee-vlan` traffic, giving highest forwarding-class priority to traffic destined for the corporate subnet, copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set firewall family inet filter egress-router-corp-class term corp-
expedite from destination-address 192.0.2.16/28
                        set firewall family inet filter egress-router-corp-class term corp-
expedite then forwarding-class expedited-forwarding
                        set firewall family inet filter egress-router-corp-class term corp-
expedite then loss-priority low
                        set firewall family inet filter egress-router-corp-class term not-to-
corp then accept
                        set interfaces ge-0/1/0 description "filter at egress router to
expedite destined for corporate network"
                        set ge-0/1/0 unit 0 family inet address 203.0.113.0
                        set interfaces ge-0/1/0 unit 0 family inet filter output egress-router-
corp-class
```

Step-by-Step Procedure

To configure and apply a firewall filter to a routed port (Layer 3 uplink module) to give highest priority to `employee-vlan` traffic destined for the corporate subnet:

1. Define the firewall filter `egress-router-corp-class`:

```
[edit]
user@switch# set firewall family inet filter egress-router-corp-class
```

2. Define the term `corp-expedite`:

```
[edit firewall]
user@switch# set family inet filter egress-router-corp-class term corp-expedite from
destination-address 192.0.2.16/28
user@switch# set family inet filter egress-router-corp-class term corp-expedite then
forwarding-class expedited-forwarding
user@switch# set family inet filter egress-router-corp-class term corp-expedite then loss-
priority low
```

3. Define the term `not-to-corp`:

```
[edit firewall]
user@switch# set family inet filter egress-router-corp-class term not-to-corp then
accept
```

4. Apply the firewall filter `egress-router-corp-class` as an output filter for the port on the switch's uplink module, which provides a Layer 3 connection to a router:

```
[edit interfaces]
user@switch# set ge-0/1/0 description "filter at egress router to expedite employee traffic
destined for corporate network"
user@switch# set ge-0/1/0 unit 0 family inet address 203.0.113.0
user@switch# set ge-0/1/0 unit 0 family inet filter output egress-router-corp-
class
```

Results

Display the results of the configuration:

```
user@switch# show
    firewall {
```

```
        family inet {
            filter egress-router-corp-class {
                term corp-expedite {
                    from {
                        destination-address 192.0.2.16/28;
                    }
                    then {
                        forwarding-class expedited-forwarding;
                        loss-priority low;
                    }
                }
                term not-to-corp {
                    then {
                        accept;
                    }
                }
            }
        }
    }
    interfaces {
        ge-0/1/0 {
            unit 0 {
                description "filter at egress router interface to expedite employee traffic
destined for corporate network";
                family inet {
                    source-address 203.0.113.0
                    filter {
                        output egress-router-corp-class;
                    }
                }
            }
        }
    }
```

Verification

IN THIS SECTION

- Verifying that Firewall Filters and Policers are Operational | **1677**
- Verifying that Schedulers and Scheduler-Maps are Operational | **1678**

To confirm that the firewall filters are working properly, perform the following tasks:

Verifying that Firewall Filters and Policers are Operational

Purpose

Verify the operational state of the firewall filters and policers that are configured on the switch.

Action

Use the operational mode command:

```
user@switch>                                show
firewall
Filter: ingress-port-voip-class-limit-tcp-icmp
Counters:
Name                                                Packets
icmp-counter                                              0
tcp-counter                                               0
Policers:
Name                                                Packets
icmp-connection-policer                                   0
tcp-connection-policer                                    0

Filter: ingress-vlan-rogue-block

Filter: egress-vlan-watch-employee
Counters:
Name                                                Packets
employee-web–counter                                      0
```

Meaning

The `show firewall` command displays the names of the firewall filters, policers, and counters that are configured on the switch. The output fields show byte and packet counts for all configured counters and the packet count for all policers.

Verifying that Schedulers and Scheduler-Maps are Operational

Purpose

Verify that schedulers and scheduler-maps are operational on the switch.

Action

Use the operational mode command:

```
user@switch>                              show class-of-service scheduler-
map

Scheduler map: default, Index: 2

  Scheduler: default-be, Forwarding class: best-effort, Index: 20
    Transmit rate: 95 percent, Rate Limit: none, Buffer size: 95 percent,
    Priority: low
    Drop profiles:
      Loss priority   Protocol    Index    Name
      Low             non-TCP         1    default-drop-profile
      Low             TCP             1    default-drop-profile
      High            non-TCP         1    default-drop-profile
      High            TCP             1    default-drop-profile

  Scheduler: default-nc, Forwarding class: network-control, Index: 22
    Transmit rate: 5 percent, Rate Limit: none, Buffer size: 5 percent,
    Priority: low
    Drop profiles:
      Loss priority   Protocol    Index    Name
      Low             non-TCP         1    default-drop-profile
      Low             TCP             1    default-drop-profile
      High            non-TCP         1    default-drop-profile
      High            TCP             1    default-drop-profileScheduler map: ethernet-diffsrv-
cos-map, Index: 21657

  Scheduler: best-effort, Forwarding class: best-effort, Index: 61257
    Transmit rate: remainder, Rate Limit: none, Buffer size: 75 percent,
    Priority: low
    Drop profiles:
      Loss priority   Protocol    Index    Name
      Low             non-TCP         1    <default-drop-profile>
```

```
    Low             TCP             1    <default-drop-profile>
    High            non-TCP         1    <default-drop-profile>
    High            TCP             1    <default-drop-profile>

Scheduler: voice-high, Forwarding class: expedited-forwarding, Index: 3123
  Transmit rate: remainder, Rate Limit: none, Buffer size: 15 percent,
  Priority: high
  Drop profiles:
    Loss priority   Protocol    Index    Name
    Low             non-TCP         1    <default-drop-profile>
    Low             TCP             1    <default-drop-profile>
    High            non-TCP         1    <default-drop-profile>
    High            TCP             1    <default-drop-profile>

Scheduler: net-control, Forwarding class: network-control, Index: 2451
  Transmit rate: remainder, Rate Limit: none, Buffer size: 10 percent,
  Priority: high
  Drop profiles:
    Loss priority   Protocol    Index    Name
    Low             non-TCP         1    <default-drop-profile>
    Low             TCP             1    <default-drop-profile>
    High            non-TCP         1    <default-drop-profile>
    High            TCP             1    <default-drop-profile>
```

Meaning

Displays statistics about the configured schedulers and schedulers-maps.

RELATED DOCUMENTATION

Example: Configuring CoS on EX Series Switches

Configuring Firewall Filters (CLI Procedure) | **1638**

Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**

Example: Configuring a Firewall Filter on a Management Interface on an EX Series Switch

IN THIS SECTION

- Requirements | 1680
- Overview and Topology | 1680
- Configuration | 1681
- Verification | 1684

You can configure a firewall filter on a management interface on an EX Series switch to filter ingress or egress traffic on the management interface on the switch. You can use utilities such as SSH or Telnet to connect to the management interface over the network and then use management protocols such as SNMP to gather statistical data from the switch.

This example discusses how to configure a firewall filter on a management interface to filter SSH packets egressing from an EX Series switch:

Requirements

This example uses the following hardware and software components:

- One EX Series switch and one management PC
- Junos OS Release 10.4 or later for EX Series switches

Overview and Topology

IN THIS SECTION

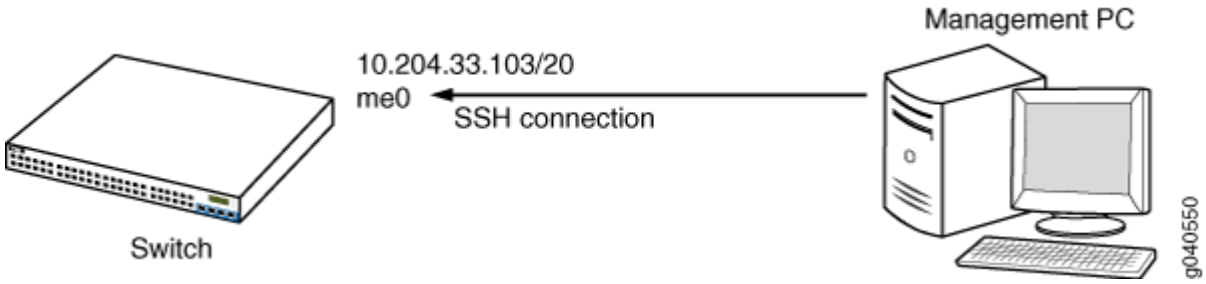
- Topology | 1681

Topology

In this example, a management PC establishes an SSH connection with the management interface on a switch to remotely manage the switch. The IP address configured for the management interface is 10.204.33.103/20. A firewall filter is configured on the management interface to count the number of packets egressing from a source SSH port on the management interface. When the management PC establishes the SSH session with the management interface, the management interface returns SSH packets to the management PC to confirm that the session is established. These SSH packets are filtered based on the match condition specified in the firewall filter before they are forwarded to the management PC. As these packets are generated from the source SSH port on the management interface, they fulfill the match condition specified for the management interface. The number of matched SSH packets provides a count of the number of packets that have traversed the management interface. A system administrator can use this information to monitor the management traffic and take any action if required.

Figure 73 on page 1681 shows the topology for this example in which a management PC establishes an SSH connection with the switch.

Figure 73: SSH Connection From a Management PC to an EX Series Switch



Configuration

IN THIS SECTION

- | 1682

To configure a firewall filter on a management interface, perform these tasks:

CLI Quick Configuration

To quickly create and configure a firewall filter on the management interface to filter SSH packets egressing from the management interface, copy the following commands and paste them into the switch terminal window:

```
[edit]
set firewall family inet filter mgmt_fil1 term t1 from source-port ssh
set firewall family inet filter mgmt_fil1 term t1 then count c1
set firewall family inet filter mgmt_fil1 term t2 then accept
set interfaces me0 unit 0 family inet filter output mgmt_fil1
```

Step-by-Step Procedure

To configure a firewall filter on the management interface to filter SSH packets:

1. Configure the firewall filter that matches SSH packets from the source port:

   ```
   [edit]
   user@switch# set firewall family inet filter (Firewall Filters) mgmt_fil1 term t1 from source-port ssh
   user@switch# set firewall family inet filter mgmt_fil1 term t1 then count c1
   user@switch# set firewall family inet filter mgmt_fil1 term t2 then accept
   ```

   These statements set a counter **c1** to count the number of SSH packets that egress from the source SSH interface on the management interface.

2. Set the firewall filter for the management interface:

   ```
   [edit]
   user@switch# set interfaces me0 unit 0 family inet filter output mgmt_fil1
   ```

   NOTE: You can also set the firewall filter for a VME interface.

Results

Check the results of the configuration:

```
[edit]
user@switch# show

interfaces {
    me0 {
        unit 0 {
            family inet {
                filter {
                    output mgmt_fil1;
                    }
                address 10.93.54.6/24;
            }
        }
    }
}

firewall {
    family inet {
        filter mgmt_fil1{
            term t1 {
                 from {
                    source-port ssh;
                 then count c1;
                }
            }
            term t2 {
                then accept;
            }
        }
    }
}
```

Verification

IN THIS SECTION

- Verifying That the Firewall Filter Is Configured on a Management Interface | 1684

To confirm that the configuration is working properly, perform these tasks:

Verifying That the Firewall Filter Is Configured on a Management Interface

Purpose

Verify that the firewall filter has been enabled on the management interface on the switch.

Action

1. Verify that the firewall filter is applied to the management interface:

```
[edit]
user@switch# show interfaces me0
unit 0 {
    family inet {
        filter {
            output mgmt_fil1;
               }
        address 10.204.33.103/20;
                }
       }
```

2. Check the counter value that is associated with the firewall filter:

```
user@switch> show firewall
Filter: mgmt_fil1
Counters:
Name                                Bytes              Packets
c1                                  0                    0
```

3. From the management PC, establish a secure shell session with the switch:

```
[user@management-pc ~]$ ssh user@10.204.33.103
```

4. Check counter values after SSH packets are generated from the switch in response to the secure shell session request by the management PC:

```
user@switch> show firewall
Filter: mgmt_fil1
Counters:
Name                                Bytes              Packets
c1                                  3533                 23
```

Meaning

The output indicates that the firewall filter has been applied to the management interface and the counter value indicates that 23 SSH packets were generated from the switch.

RELATED DOCUMENTATION

Configuring Firewall Filters (CLI Procedure) | 1638

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device

IN THIS SECTION

- Requirements | 1686
- Overview and Topology | 1686
- Configuration | 1686
- Verification | 1690

This example describes how to set up filter-based forwarding on EX Series switches or a QFX10000. You can configure filter-based forwarding by using a firewall filter to forward matched traffic to a specific virtual routing instance.

Requirements

This example applies to both EX Series switches running Junos OS Release 9.4 or later, and QFX10000 switches running Junos OS Release 15.1X53-D10 or later.

Overview and Topology

In this example, we create a firewall filter to match traffic being sent from one application server to another according to the destination address (192.168.0.1) of packets egressing the source application server. Matching packets are routed to a virtual routing instance which forwards the traffic to a security device, which then forwards the traffic on to the destination application server.

NOTE: Filter-based forwarding does not work with IPv6 interfaces on some Juniper switches.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1686**
- Procedure | **1687**
- Results | **1688**

To configure filter-based forwarding:

CLI Quick Configuration

To use this example on your own device, copy the following commands into a text file, remove the line breaks, and change the necessary details to fit your configuration. Then copy and paste the commands into your CLI at the `[edit]` hierarchy level.

```
[edit]
                    set interfaces xe-0/0/0 unit 0 family inet address
10.1.0.1/24
                    set interfaces xe-0/0/3 unit 0 family inet address
```

```
10.1.3.1/24
                    set firewall family inet filter f1 term t1 from source-address
10.1.0.50/32
                    set firewall family inet filter f1 term t1 from protocol
tcp
                    set interfaces xe-0/0/0 unit 0 family inet filter input f1
                    set routing-instances vrf01 instance-type virtual-router
                    set routing-instances vrf01 interface xe-0/0/3.0
                    set routing-instances vrf01 routing-options static route 192.168.0.1/24
next-hop 10.1.3.254
                    set firewall family inet filter f1 term t1 then routing-instance
vrf01
```

Procedure

Step-by-Step Procedure

To configure filter-based forwarding:

1. Configure an interface to connect to the application server:

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 0 family inet address 10.1.0.1/24
```

2. Configure an interface to connect to the security device:

```
[edit interfaces]
user@switch# set xe-0/0/3 unit 0 family inet address 10.1.3.1/24
```

3. Create a firewall filter that matches packets based on the address of the application server that the traffic will be sent from. Also configure the filter so that it matches only TCP packets:

```
[edit firewall]
user@switch# set family inet filter f1 term t1 from source-address 10.1.0.50/32
user@switch# set firewall family inet filter f1 term t1 from protocol
tcp
```

4. Apply the filter to the interface that connects to the source application server and configure it to match incoming packets:

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 0 family inet filter input f1
```

5. Create a virtual router:

```
[edit]
user@switch# set routing-instances vrf01 instance-type virtual-router
```

6. Associate the virtual router with the interface that connects to the security device:

```
[edit routing-instances]
user@switch# set vrf01 interface xe-0/0/3.0
```

7. Configure the routing information for the virtual routing instance:

```
[edit routing-instances]
user@switch# set vrf01 routing-options static route 192.168.0.1/24 next-hop 10.1.3.254
```

8. Set the filter to forward packets to the virtual router:

```
[edit firewall]
user@switch# set family inet filter f1 term t1 then routing-instance
vrf01
```

Results

Check the results of the configuration:

```
user@switch> show configuration
interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                filter {
```

```
                    input f1;
                }
                address 10.1.0.1/24;
            }
        }
    }
    xe-0/0/3 {
        unit 0 {
            family inet {
                address 10.1.3.1/24;
            }
        }
    }
}
firewall {
    family inet {
        filter f1 {
            term t1 {
                from {
                    source-address {
                        10.1.0.50/32;
                    }
                    protocol tcp;
                }
                then {
                    routing-instance vrf01;
                }
            }
        }
    }
}
    routing-instances {
        vrf01 {
            instance-type virtual-router;
            interface xe-0/0/3.0;
            routing-options {
                static {
                    route 192.168.0.1/24 next-hop 10.1.3.254;
                }
            }
        }
    }
```

Verification

IN THIS SECTION

- Verifying That Filter-Based Forwarding Was Configured | **1690**

To confirm that the configuration is working properly, perform these tasks:

Verifying That Filter-Based Forwarding Was Configured

Purpose

Verify that filter-based forwarding was properly enabled on the switch.

Action

1. Use the `show interfaces filters` command:

```
user@switch>                                    show interfaces filters
xe-0/0/0.0
Interface        Admin Link Proto Input Filter        Output Filter
xe-0/0/0.0       up    down inet  fil
```

2. Use the `show route forwarding-table` command:

```
user@switch> show route forwarding-table

Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            user     1 0:12:f2:21:cf:0    ucst   331     4 me0.0
default            perm     0                    rjct    36     3
0.0.0.0/32         perm     0                    dscd    34     1
10.1.0.0/24        ifdn     0                    rslv   613     1 xe-0/0/0.0
10.1.0.0/32        iddn     0 10.1.0.0           recv   611     1 xe-0/0/0.0
10.1.0.1/32        user     0                    rjct    36     3
10.1.0.1/32        intf     0 10.1.0.1           locl   612     2
```

```
10.1.0.1/32        iddn     0 10.1.0.1          locl   612     2
10.1.0.255/32      iddn     0 10.1.0.255        bcst   610     1 xe-0/0/0.0
10.1.1.0/26        ifdn     0                   rslv   583     1 vlan.0
10.1.1.0/32        iddn     0 10.1.1.0          recv   581     1 vlan.0
10.1.1.1/32        user     0                   rjct    36     3
10.1.1.1/32        intf     0 10.1.1.1          locl   582     2
10.1.1.1/32        iddn     0 10.1.1.1          locl   582     2
10.1.1.63/32       iddn     0 10.1.1.63         bcst   580     1 vlan.0
255.255.255.255/32 perm     0                   bcst    32     1

Routing table: vrf01.inet
Internet:
Destination        Type RtRef Next hop          Type Index NhRef Netif
default            perm     0                   rjct   559     2
0.0.0.0/32         perm     0                   dscd   545     1
10.1.3.0/24        ifdn     0                   rslv   617     1 xe-0/0/3.0
10.1.3.0/32        iddn     0 10.1.3.0          recv   615     1 xe-0/0/3.0
10.1.3.1/32        user     0                   rjct   559     2
192.168.0.1/24     user     0 10.1.3.254        ucst   616     2 xe-0/0/3.0
192.168.0.1/24     user     0 10.1.3.254        ucst   616     2 xe-0/0/3.0
10.1.3.255/32      iddn     0 10.1.3.255        bcst   614     1 xe-0/0/3.0
224.0.0.0/4        perm     0                   mdsc   546     1
224.0.0.1/32       perm     0 224.0.0.1         mcst   529     1
255.255.255.255/32 perm     0                   bcst   543     1

Routing table: default.iso
ISO:
Destination        Type RtRef Next hop          Type Index NhRef Netif
default            perm     0                   rjct    60     1

Routing table: vrf01.iso
ISO:
Destination        Type RtRef Next hop          Type Index NhRef Netif
default            perm     0                   rjct   600     1
```

Meaning

The output indicates that the filter was created on the interface and that the virtual routing instance is forwarding matching traffic to the correct IP address.

RELATED DOCUMENTATION

Configuring Firewall Filters | 1826

Understanding Filter-Based Forwarding | 1854

Understanding Virtual Router Routing Instances

Example: Applying Firewall Filters to Multiple Supplicants on Interfaces Enabled for 802.1X or MAC RADIUS Authentication

IN THIS SECTION

- Requirements | 1692
- Overview and Topology | 1693
- Configuration | 1695
- Verification | 1698

On EX Series switches, firewall filters that you apply to interfaces enabled for 802.1X or MAC RADIUS authentication are dynamically combined with the per-user policies sent to the switch from the RADIUS server. The switch uses internal logic to dynamically combine the interface firewall filter with the user policies from the RADIUS server and create an individualized policy for each of the multiple users or nonresponsive hosts that are authenticated on the interface.

This example describes how dynamic firewall filters are created for multiple supplicants on an 802.1X-enabled interface (the same principles shown in this example apply to interfaces enabled for MAC RADIUS authentication):

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.5 or later for EX Series switches
- One EX Series switch
- One RADIUS authentication server. The authentication server acts as the backend database and contains credential information for hosts (supplicants) that have permission to connect to the network.

Before you apply firewall filters to an interface for use with multiple supplicants, be sure you have:

- Set up a connection between the switch and the RADIUS server. See Example: Connecting a RADIUS Server for 802.1X to an EX Series Switch.
- Configured 802.1X authentication on the switch, with the authentication mode for interface **ge-0/0/2** set to **multiple**. See Configuring 802.1X Interface Settings (CLI Procedure) and Example: Setting Up 802.1X for Single-Supplicant or Multiple-Supplicant Configurations on an EX Series Switch.
- Configured users on the RADIUS authentication server.

Overview and Topology

IN THIS SECTION

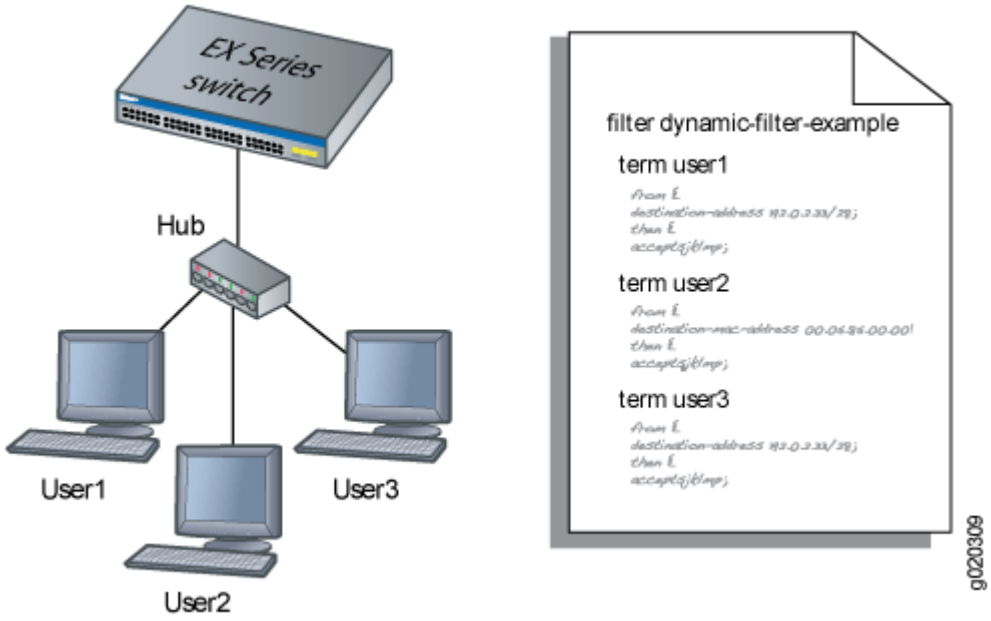
- Topology | **1693**

Topology

When the 802.1X configuration on an interface is set to multiple supplicant mode, the system dynamically combines interface firewall filter with the user policies sent to the switch from the RADIUS server during authentication and creates separate terms for each user. Because there are separate terms for each user authenticated on the interface, you can, as shown in this example, use counters to view the activities of individual users that are authenticated on the same interface.

When a new user (or a nonresponsive host) is authenticated on an interface, the system adds a term to the firewall filter associated with the interface, and the term (policy) for each user is associated with the MAC address of the user. The term for each user is based on the user-specific filters set on the RADIUS server and the filters configured on the interface. For example, as shown in Figure 74 on page 1694, when User1 is authenticated by the EX Series switch, the system creates the firewall filter **dynamic-filter-example**. When User2 is authenticated, another term is added to the firewall filter, and so on.

Figure 74: Conceptual Model: Dynamic Filter Updated for Each New User

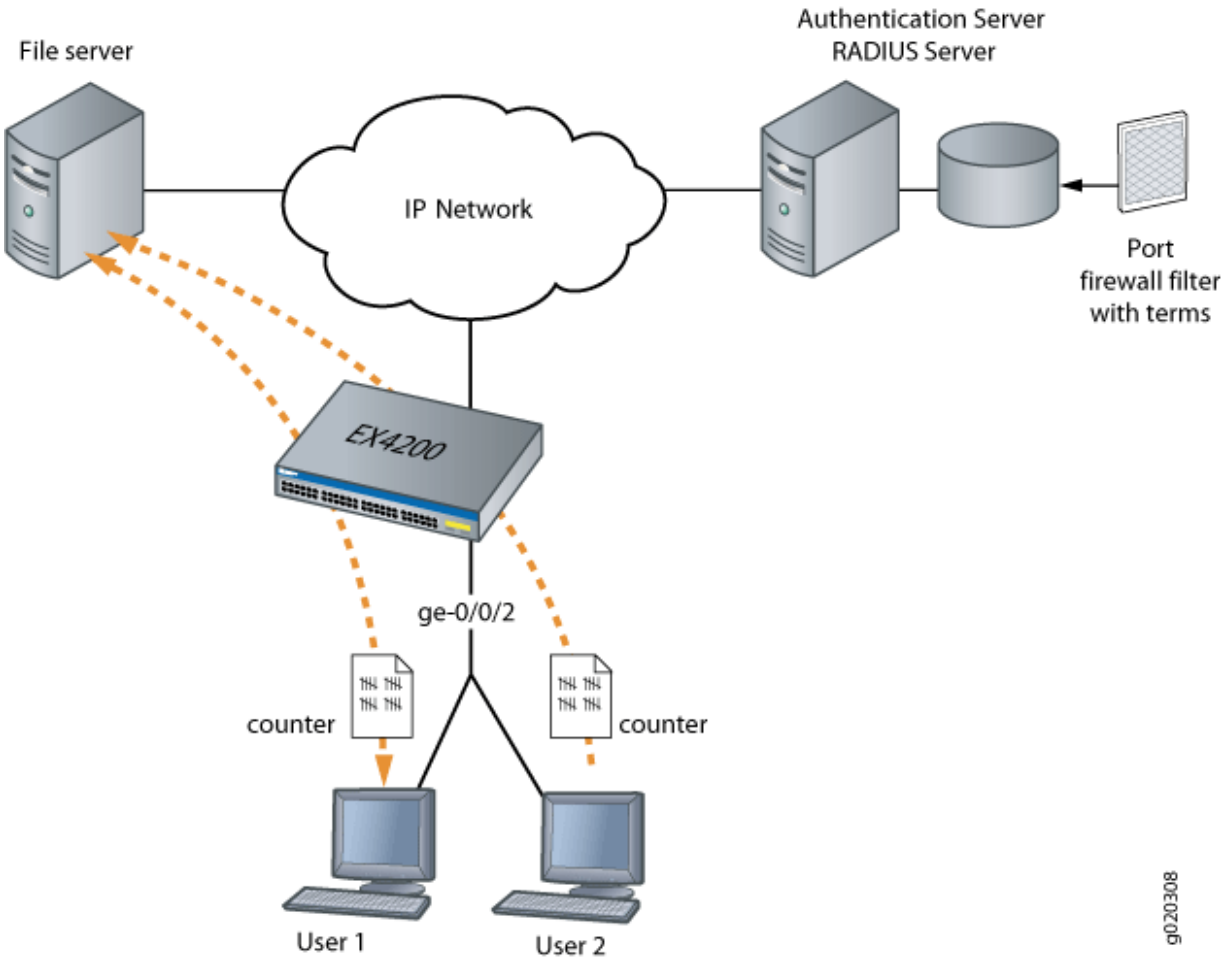


This is a conceptual model of the internal process—you cannot access or view the dynamic filter.

> **NOTE**: If the firewall filter on the interface is modified after the user (or nonresponsive host) is authenticated, the modifications are not reflected in the dynamic filter unless the user is reauthenticated.

In this example, you configure a firewall filter to count the requests made by each endpoint authenticated on interface **ge-0/0/2** to the file server, which is located on subnet **192.0.2.16/28**, and set policer definitions to rate limit the traffic. Figure 75 on page 1695 shows the network topology for this example.

Figure 75: Multiple Supplicants on an 802.1X-Enabled Interface Connecting to a File Server



Configuration

IN THIS SECTION

- Configuring Firewall Filters on Interfaces with Multiple Supplicants | **1696**

To configure firewall filters for multiple supplicants on 802.1X-enabled interfaces:

Configuring Firewall Filters on Interfaces with Multiple Supplicants

CLI Quick Configuration

To quickly configure firewall filters for multiple supplicants on an 802.1X-enabled interface copy the following commands and paste them into the switch terminal window:

```
[edit]
                        set protocols dot1x authenticator interface ge-0/0/2 supplicant
multiple
                        set firewall family ethernet-switching filter filter1 term term1 from
destination-address 192.0.2.16/28
set firewall policer p1 if-exceeding bandwidth-limit 1m
set firewall policer p1 if-exceeding burst-size-limit 1k
                        set firewall family ethernet-switching filter filter1 term term1 then
count counter1
set firewall family ethernet-switching filter filter1 term term2 then policer p1
```

Step-by-Step Procedure

To configure firewall filters on an interface enabled for multiple supplicants:

1. Configure interface **ge-0/0/2** for multiple supplicant mode authentication:

```
[edit protocols dot1x]
user@switch# set authenticator interface ge-0/0/2 supplicant multiple
```

2. Set policer definition:

```
user@switch# show policer p1 |display set
set firewall policer p1 if-exceeding bandwidth-limit 1m
set firewall policer p1 if-exceeding burst-size-limit 1k
set firewall policer p1 then discard
```

3. Configure a firewall filter to count packets from each user and a policer that limits the traffic rate. As each new user is authenticated on the multiple supplicant interface, this filter term will be included in the dynamically created term for the user:

```
[edit firewall family ethernet-switching]
user@switch# set filter filter1 term term1 from destination-address 192.0.2.16/28
user@switch# set filter filter1 term term1 then count counter1
user@switch# set filter filter1 term term2 then policer p1
```

Results

Check the results of the configuration:

```
user@switch> show configuration

firewall {
    family ethernet-switching {
        filter filter1 {
            term term1 {
                from {
                    destination-address {
                        192.0.2.16/28;
                    }
                }
                then count counter1;
                term term2 {
                from {
                    destination-address {
                        192.0.2.16/28;
                    }
                }
                then policer p1;
            }
        }
    }
    policer p1 {
        if-exceeding {
            bandwidth-limit 1m;
                burst-size-limit 1k;
            }
            then discard;
```

```
        }
}
protocols {
    dot1x {
        authenticator
            interface ge-0/0/2 {
                supplicant multiple;
            }
        }
    }
```

Verification

IN THIS SECTION

- Verifying Firewall Filters on Interfaces with Multiple Supplicants | **1698**

To confirm that the configuration is working properly, perform these tasks:

Verifying Firewall Filters on Interfaces with Multiple Supplicants

Purpose

Verify that firewall filters are functioning on the interface with multiple supplicants.

Action

1. Check the results with one user authenticated on the interface. In this case, the user is authenticated on **ge-0/0/2**:

```
user@switch> show dot1x firewall

Filter: dot1x_ge-0/0/2
Counters
counter1_dot1x_ge-0/0/2_user1 100
```

2. When a second user, User2, is authenticated on the same interface, **ge-0/0/2**, you can verify that the filter includes the results for both of the users authenticated on the interface:

```
user@switch>  show dot1x firewall

Filter: dot1x-filter-ge-0/0/0
Counters
counter1_dot1x_ge-0/0/2_user1 100
counter1_dot1x_ge-0/0/2_user2 400
```

Meaning

The results displayed by the `show dot1x firewall` command output reflect the dynamic filter created with the authentication of each new user. User1 accessed the file server located at the specified destination address 100 times, while User2 accessed the same file server 400 times.

RELATED DOCUMENTATION

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Filtering 802.1X Supplicants by Using RADIUS Server Attributes

Verifying That Policers Are Operational

IN THIS SECTION

- Purpose | 1699
- Action | 1700
- Meaning | 1700

Purpose

After you configure policers and include them in firewall filter configurations, you can perform the following tasks to verify that the policers configured on EX Series switches are working properly.

Action

Use the operational mode command to verify that the policers on the switch are working properly:

```
user@switch>                        show policer
Filter: egress-vlan-watch-employee
Filter: ingress-port-filter
Filter: ingress-port-voip-class-limit-tcp-icmp
Policers:
Name                                                Packets
icmp-connection-policer                                   0
tcp-connection-policer                                    0
Filter: ingress-vlan-rogue-block
Filter: ingress-vlan-limit-guest
```

Meaning

The `show policer` command displays the names of all firewall filters and policers that are configured on the switch. For each policer that is specified in a filter configuration, the output field shows the current packet count for all packets that exceed the specified rate limits.

RELATED DOCUMENTATION

- Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**
- Configuring Firewall Filters (CLI Procedure) | **1638**
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**
- Monitoring Firewall Filter Traffic | **1867**

Troubleshooting Firewall Filters

IN THIS SECTION

- Troubleshooting QFX10000 Switches | **1701**
- Troubleshooting Other Switches | **1702**

Use the following information to troubleshoot your firewall filter configuration.

Troubleshooting QFX10000 Switches

IN THIS SECTION

- Do Not Combine Match Conditions for Different Layers | 1701
- Layer 2 Packets Cannot be Discarded with Firewall Filters | 1701
- Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces | 1702

This section describes issues specific to QFX10000 switches:

Do Not Combine Match Conditions for Different Layers

On QFX10000 switches, do not combine match conditions for Layer 2 and any other layer in a `family ethernet-switching` filter. (For example, do not include conditions that match MAC addresses and IP addresses in the same filter.) If you do so, the filter will commit successfully but will not work. You will also see the following log message: `L2 filter` *`filter-name`* `doesn't support mixed L2 and L3/L4 match conditions. Please re-config.`

Layer 2 Packets Cannot be Discarded with Firewall Filters

IN THIS SECTION

- Problem | 1701
- Solution | 1702

Problem

Description

Layer 2 (L2) control packets such as Link Layer Discovery Protocol (LLDP) and bridge protocol data unit (BPDU) cannot be discarded with firewall filters.

Solution

Configure distributed denial-of-service (DDoS) protection on the L2 control packet and set the aggregate policer bandwidth and burst values to the minimum value of 1. For example,

```
[edit system ddos-protection protocols protocol name]
```

user@host# **set aggregate bandwidth 1**

```
[edit system ddos-protection protocols protocol name]
```

user@host# **set aggregate burst 1**

Protect-RE (loopback) Firewall Filter Does Not Filter Packets Applied to EM0 Interfaces

IN THIS SECTION

- Problem | **1702**
- Solution | **1702**

Problem

Description

On QFX10000 switches, the Protect-RE (loopback) firewall filter does not filter packets applied to EM0 interfaces including SNMP, Telnet, and other services.

Solution

This is expected behavior.

Troubleshooting Other Switches

IN THIS SECTION

- Firewall Filter Configuration Returns a No Space Available in TCAM Message | **1703**

- Filter Counts Previously Dropped Packet | **1706**
- Matching Packets Not Counted | **1707**
- Counter Reset When Editing Filter | **1708**
- Cannot Include loss-priority and policer Actions in Same Term | **1708**
- Cannot Egress Filter Certain Traffic Originating on QFX Switch | **1709**
- Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **1710**
- Egress Firewall Filters with Private VLANs | **1710**
- Egress Filtering of L2PT Traffic Not Supported | **1712**
- Cannot Drop BGP Packets in Certain Circumstances | **1712**
- Invalid Statistics for Policer | **1713**
- Policers can Limit Egress Filters | **1713**

This section describes issues specific to QFX switches other than QFX10000 switches. This information also applies to OCX1100 switches and EX4600 switches.

Firewall Filter Configuration Returns a No Space Available in TCAM Message

IN THIS SECTION

- Problem | **1703**
- Solution | **1704**

Problem

Description

When a firewall filter configuration exceeds the amount of available Ternary Content Addressable Memory (TCAM) space, the system returns the following `syslogd` message:

```
No space available in tcam.
Rules for filter filter-name will not be installed.
```

A switch returns this message during the commit operation if the firewall filter that has been applied to a port, VLAN, or Layer 3 interface exceeds the amount of space available in the TCAM table. The filter is not applied, but the commit operation for the firewall filter configuration is completed in the CLI module.

Solution

When a firewall filter configuration exceeds the amount of available TCAM table space, you must configure a new firewall filter with fewer filter terms so that the space requirements for the filter do not exceed the available space in the TCAM table.

You can perform either of the following procedures to correct the problem:

To delete the filter and its binding and apply the new smaller firewall filter to the same binding:

1. Delete the filter and its binding to ports, VLANs, or Layer 3 interfaces. For example:

```
[edit]
user@switch# delete firewall family ethernet-switching filter ingress-vlan-rogue-block
user@switch# delete vlans employee-vlan description "filter to block rogue devices on
employee-vlan"
user@switch# delete vlans employee-vlan filter input ingress-vlan-rogue-block
```

2. Commit the changes:

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

3. Configure a smaller filter with fewer terms that does not exceed the amount of available TCAM space. For example:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block ...
```

4. Apply (bind) the new firewall filter to a port, VLAN , or Layer 3 interface. For example:

```
[edit]
user@switch# set vlans employee-vlan description "filter to block rogue devices on employee-
vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

5. Commit the changes:

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

To apply a new firewall filter and overwrite the existing binding but not delete the original filter:

1. Configure a firewall filter with fewer terms than the original filter:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block...
```

2. Apply the firewall filter to the port, VLAN, or Layer 3 interfaces to overwrite the binding of the original filter—for example:

```
[edit]
user@switch# set vlans employee-vlan description "smaller filter to block rogue devices on
employee-vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

   Because you can apply no more than one firewall filter per VLAN per direction, the binding of the original firewall filter to the VLAN is overwritten with the new firewall filter `new-ingress-vlan-rogue-block`.

3. Commit the changes:

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

NOTE: The original filter is not deleted and is still available in the configuration.

Filter Counts Previously Dropped Packet

IN THIS SECTION

- Problem | **1706**
- Solution | **1707**

Problem

Description

If you configure two or more filters in the same direction for a physical interface and one of the filters includes a counter, the counter will be incorrect if the following circumstances apply:

- You configure the filter that is applied to packets first to discard certain packets. For example, imagine that you have a VLAN filter that accepts packets sent to 10.10.1.0/24 addresses and implicitly discards packets sent to any other addresses. You apply the filter to the `admin` VLAN in the output direction, and interface xe-0/0/1 is a member of that VLAN.
- You configure a subsequent filter to accept and count packets that are dropped by the first filter. In this example, you have a port filter that accepts and counts packets sent to 192.168.1.0/24 addresses that is also applied to xe-0/0/1 in the output direction.

The egress VLAN filter is applied first and correctly discards packets sent to 192.168.1.0/24 addresses. The egress port filter is applied next and counts the discarded packets as matched packets. The packets are not forwarded, but the counter displayed by the egress port filter is incorrect.

Remember that the order in which filters are applied depends on the direction in which they are applied, as indicated here:

Ingress filters:

1. Port (Layer 2) filter
2. VLAN filter
3. Router (Layer 3) filter

Egress filters:

1. Router (Layer 3) filter
2. VLAN filter

3. Port (Layer 2) filter

Solution

This is expected behavior.

Matching Packets Not Counted

IN THIS SECTION

- Problem | 1707
- Solution | 1707

Problem

Description

If you configure two egress filters with counters for a physical interface and a packet matches both of the filters, only one of the counters includes that packet.

For example:

- You configure an egress port filter with a counter for interface xe-0/0/1.
- You configure an egress VLAN filter with a counter for the `admin`VLAN, and interface xe-0/0/1 is a member of that VLAN.
- A packet matches both filters.

In this case, the packet is counted by only one of the counters even though it matched both filters.

Solution

This is expected behavior.

Counter Reset When Editing Filter

IN THIS SECTION

- Problem | 1708
- Solution | 1708

Problem

Description

If you edit a firewall filter term, the value of any counter associated with any term in the same filter is set to 0, including the implicit counter for any policer referenced by the filter. Consider the following examples:

- Assume that your filter has `term1`, `term2`, and `term3`, and each term has a counter that has already counted matching packets. If you edit any of the terms in any way, the counters for all the terms are reset to 0.
- Assume that your filter has `term1` and `term2`. Also assume that `term2` has a `policer` action modifier and the implicit counter of the policer has already counted 1000 matching packets. If you edit `term1` or `term2` in any way, the counter for the policer referenced by `term2` is reset to 0.

Solution

This is expected behavior.

Cannot Include loss-priority and policer Actions in Same Term

IN THIS SECTION

- Problem | 1709
- Solution | 1709

Problem

Description

You cannot include both of the following actions in the same firewall filter term in a QFX Series switch:

- `loss-priority`
- `policer`

If you do so, you see the following error message when you attempt to commit the configuration: “cannot support policer action if loss-priority is configured.”

Solution

This is expected behavior.

Cannot Egress Filter Certain Traffic Originating on QFX Switch

IN THIS SECTION

- Problem | **1709**
- Solution | **1709**

Problem

Description

On a QFX Series switch, you cannot filter certain traffic with a firewall filter applied in the output direction if the traffic originates on the QFX switch. This limitation applies to control traffic for protocols such as ICMP (ping), STP, LACP, and so on.

Solution

This is expected behavior.

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling

IN THIS SECTION

- Problem | 1710
- Solution | 1710

Problem

Description

If you create a firewall filter that includes a match condition of `dot1q-tag` or `dot1q-user-priority` and apply the filter on input to a trunk port that participates in a service VLAN, the match condition does not work if the Q-in-Q EtherType is not 0x8100. (When Q-in-Q tunneling is enabled, trunk interfaces are assumed to be part of the service provider or data center network and therefore participate in service VLANs.)

Solution

This is expected behavior. To set the Q-in-Q EtherType to 0x8100, enter the **set dot1q-tunneling ethertype 0x8100** statement at the `[edit ethernet-switching-options]` hierarchy level. You must also configure the other end of the link to use the same Ethertype.

Egress Firewall Filters with Private VLANs

IN THIS SECTION

- Problem | 1711
- Solution | 1711

Problem

Description

If you apply a firewall filter in the output direction to a primary VLAN, the filter also applies to the secondary VLANs that are members of the primary VLAN when the traffic egresses with the primary VLAN tag or isolated VLAN tag, as listed below:

- Traffic forwarded from a secondary VLAN trunk port to a promiscuous port (trunk or access)
- Traffic forwarded from a secondary VLAN trunk port that carries an isolated VLAN to a PVLAN trunk port.
- Traffic forwarded from a promiscuous port (trunk or access) to a secondary VLAN trunk port
- Traffic forwarded from a PVLAN trunk port. to a secondary VLAN trunk port
- Traffic forwarded from a community port to a promiscuous port (trunk or access)

If you apply a firewall filter in the output direction to a primary VLAN, the filter does *not* apply to traffic that egresses with a community VLAN tag, as listed below:

- Traffic forwarded from a community trunk port to a PVLAN trunk port
- Traffic forwarded from a secondary VLAN trunk port that carries a community VLAN to a PVLAN trunk port
- Traffic forwarded from a promiscuous port (trunk or access) to a community trunk port
- Traffic forwarded from a PVLAN trunk port. to a community trunk port

If you apply a firewall filter in the output direction to a community VLAN, the following behaviors apply:

- The filter is applied to traffic forwarded from a promiscuous port (trunk or access) to a community trunk port (because the traffic egresses with the community VLAN tag).
- The filter is applied to traffic forwarded from a community port to a PVLAN trunk port (because the traffic egresses with the community VLAN tag).
- The filter is *not* applied to traffic forwarded from a community port to a promiscuous port (because the traffic egresses with the primary VLAN tag or untagged).

Solution

These are expected behaviors. They occur only if you apply a firewall filter to a private VLAN in the output direction and do not occur if you apply a firewall filter to a private VLAN in the input direction.

Egress Filtering of L2PT Traffic Not Supported

IN THIS SECTION

- Problem | 1712
- Solution | 1712

Problem

Description

Egress filtering of L2PT traffic is not supported on the QFX3500 switch. That is, if you configure L2PT to tunnel a protocol on an interface, you cannot also use a firewall filter to filter traffic for that protocol on that interface in the output direction. If you commit a configuration for this purpose, the firewall filter is not applied to the L2PT-tunneled traffic.

Solution

This is expected behavior.

Cannot Drop BGP Packets in Certain Circumstances

IN THIS SECTION

- Problem | 1712
- Solution | 1713

Problem

Description

BGP packets with a time-to-live (TTL) value greater than 1 cannot be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface. BGP packets with TTL value of 1 or 0 can be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface.

Solution

This is expected behavior.

Invalid Statistics for Policer

IN THIS SECTION

- Problem | 1713
- Solution | 1713

Problem

Description

If you apply a single-rate two-color policer in more than 128 terms in a firewall filter, the output of the `show firewall` command displays incorrect data for the policer.

Solution

This is expected behavior.

Policers can Limit Egress Filters

IN THIS SECTION

- Problem | 1713
- Solution | 1714

Problem

Description

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry

TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

Solution

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

CHAPTER 27

Configuring Firewall Filters (QFX Series Switches, EX4600 Switches, PTX Series Routers)

IN THIS CHAPTER

- Overview of Firewall Filters (QFX Series) | **1716**
- Understanding Firewall Filter Planning | **1719**
- Planning the Number of Firewall Filters to Create | **1721**
- Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | **1731**
- Firewall Filter Match Conditions and Actions (QFX10000 Switches) | **1780**
- Firewall Filter Match Conditions and Actions (PTX Series Routers) | **1798**
- Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers | **1823**
- Configuring Firewall Filters | **1826**
- Applying Firewall Filters to Interfaces | **1835**
- Overview of MPLS Firewall Filters on Loopback Interface | **1835**
- Configuring MPLS Firewall Filters and Policers on Switches | **1837**
- Configuring MPLS Firewall Filters and Policers on Routers | **1840**
- Configuring MPLS Firewall Filters and Policers | **1849**
- Understanding How a Firewall Filter Tests a Protocol | **1852**
- Understanding Firewall Filter Processing Points for Bridged and Routed Packets | **1853**
- Understanding Filter-Based Forwarding | **1854**
- Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1855**
- Configuring a Firewall Filter to De-Encapsulate GRE or IPIP Traffic | **1862**
- Verifying That Firewall Filters Are Operational | **1866**
- Monitoring Firewall Filter Traffic | **1867**
- Troubleshooting Firewall Filter Configuration | **1871**

Overview of Firewall Filters (QFX Series)

IN THIS SECTION

- Where You Can Apply Filters | 1716
- What Makes up a Firewall Filter | 1717
- How Firewall Filters are Processed | 1718

Firewall filters, sometimes called *access control lists* (ACLs), provide rules that define whether to accept or discard packets that are transiting an interface. If a packet is accepted, you can configure more actions on the packet, such as class-of-service (CoS) marking (grouping similar types of traffic together and treating each type of traffic as a class with its own level of service priority) and traffic policing (controlling the maximum rate of traffic sent or received).

You can configure firewall filters to determine where to accept or discard a packet before it enters or exits a port, VLAN, Layer 2 CCC, Layer 3 (routed) interface, Routed VLAN interface (RVI), or MPLS interface.

An *ingress (input) firewall filter* is applied to packets that are entering an interface or VLAN, and an *egress (output)* firewall filter is applied to packets that are exiting an interface or VLAN.

NOTE: Policers on network port, layer 2 and layer 3, or IRB interfaces do not police host-bound traffic. But if you want to prevent DDoS attacks, then you can create a firewall filter on the lo0 that protects the routing engine.

Where You Can Apply Filters

After you configure the firewall filter, you can apply it to the following:

- Port—Filters Layer 2 traffic transiting system ports.
- VLAN—Filters and provides access control for Layer 2 packets that enter a VLAN, are bridged within a VLAN, or leave a VLAN.
- Layer 3 (routed) interface—Filters traffic on IPv4 and IPv6 interfaces, routed VLAN interfaces (RVI), and the loopback interface. The loopback interface filters traffic sent to the switch itself or generated by the switch.

- Layer 2 CCC interface—Filters Layer 2 circuit cross-connect (CCC) interfaces.
- MPLS—Filters MPLS interfaces.

You can also apply a firewall filter to a management interface (for example, me0) on a QFX and EX4600 standalone switch. You can't apply a filter to a management interface on a QFX3000-G or QFX3000-M system.

> **NOTE**: You can apply only one firewall filter to a port, VLAN, or Layer 2 CCC interface for a given direction. For example, for interface ge-0/0/6.0, you can apply one filter for the ingress direction and one for the egress direction.

- (QFX Series) Starting with Junos OS Release 13.2X51-D15, you can apply a filter to a loopback interface in the egress direction.
- (QFX10000) Starting with Junos OS Release 18.2R1, you can apply ingress and egress firewall filters with `count` and `discard` as policer actions on Layer 2 circuit interfaces.
- (QFX10002-36Q, QFX10002-72Q, QFX10002-60C, QFX10008, QFX10016, PTX10008, PTX10016) Starting with Junos OS Release 19.2R1, you can apply the `interface`, `forwarding-class`, and `loss-priority` match conditions in the egress direction on IPv4 and IPv6 interfaces.

> **NOTE**: The EX4600 , QFX5000 series and QFX5000 EVO series switches do not depend on the VRF match for loopback filters configured at different routing instances. Loopback filters per routing instance (such as lo0.100, lo0.103, lo0.105) are not supported and may cause unpredictable behavior. We recommend that you only apply the loopback filter (lo0.0) to the master routing instance.

What Makes up a Firewall Filter

When you configure a firewall filter, you define the family address type (ethernet-switching, inet (for IPv4), inet6 (for IPv6), circuit cross-connect (CCC), or MPLS), the filtering criteria (terms, with match conditions,) and the action to take if a match occurs.

Each term consists of the following

- Match condition—Values that a packet must contain to be considered a match. You can specify values for most fields in the IP, TCP, UDP, or ICMP headers. You can also match on interface names.
- Action—Action taken if a packet matches a match condition. You can configure a firewall filter to accept, discard, or reject a matching packet and then perform more actions, such as counting, classifying, and policing. The default action is accept.

How Firewall Filters are Processed

If there are multiple terms in a filter, the order of the terms is important. If a packet matches the first term, the switch takes the action defined by that term, and no other terms are evaluated. If the switch doesn't find a match between the packet and the first term, it compares the packet to the next term. If no match occurs between the packet and the second term, the system continues to compare the packet to each successive term in the filter until a match is found. If no terms are matched, the switch discards the packet by default.

Release History Table

Release	Description
19.2R1	(QFX10002-36Q, QFX10002-72Q, QFX10002-60C, QFX10008, QFX10016, PTX10008, PTX10016) Starting with Junos OS Release 19.2R1, you can apply the `interface`, `forwarding-class`, and `loss-priority` match conditions in the egress direction on IPv4 and IPv6 interfaces.
18.2R1	(QFX10000) Starting with Junos OS Release 18.2R1, you can apply ingress and egress firewall filters with `count` and `discard` as policer actions on Layer 2 circuit interfaces.
13.2X51-D15	(QFX Series) Starting with Junos OS Release 13.2X51-D15, you can apply a filter to a loopback interface in the egress direction.

RELATED DOCUMENTATION

- Understanding How Firewall Filters Are Evaluated | **854**
- Understanding Firewall Filter Planning | **1719**
- Planning the Number of Firewall Filters to Create | **1721**
- Configuring Firewall Filters | **1826**
- Configuring MPLS Firewall Filters and Policers | **1849**
- Overview of Policers | **2198**
- Understanding Firewall Filter Match Conditions | **856**
- Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | **1731**
- Firewall Filter Match Conditions and Actions (QFX10000 Switches) | **1780**

Understanding Firewall Filter Planning

Before you create a *firewall filter* and apply it, determine what you want the filter to accomplish and how to use its match conditions and actions to achieve your goals. It is important that you understand how packets are matched, the default and configured actions of the firewall filter, and where to apply the firewall filter.

You can apply no more than one firewall filter per port, VLAN, or router interface per direction (input and output). For example, for a given port you can apply at most one filter in the input direction and one filter in the output direction. You should try to be conservative in the number of terms (rules) that you include in each firewall filter, because a large number of terms requires longer processing time during a commit operation and can make testing and troubleshooting more difficult.

Before you configure and apply firewall filters, answer the following questions for each of them:

1. What is the purpose of the filter?

   For example, the system can drop packets based on header information, rate-limit traffic, classify packets into forwarding classes, log and count packets, or prevent denial-of-service attacks.

2. What are the appropriate match conditions? Determine the packet header fields that the packet must contain for a match. Possible fields include:

   - Layer 2 header fields—Source and destination MAC addresses, 802.1Q tag, Ethernet type, or VLAN.
   - Layer 3 header fields—Source and destination IP addresses, protocols, and IP options (IP precedence, IP fragmentation flags, or TTL type).
   - TCP header fields—Source and destination ports and flags.
   - ICMP header fields—Packet type and code.

3. What are the appropriate actions to take if a match occurs?

   The system can accept, discard, or reject packets.

4. What additional action modifiers might be required?

   For example, you can configure the system to mirror (copy) packets to a specified port, count matching packets, apply traffic management, or police packets.

5. On what port, router interface, or VLAN should the firewall filter be applied?

   Start with the following basic guidelines:

- If packets entering or leaving a Layer 2 interface (port) need to be filtered, apply the filter at the `[edit family ethernet switching filter]` hierarchy level. This is a port filter.
- If packets entering or leaving any port in a specific VLAN need to be filtered, use a VLAN filter.
- If packets entering or leaving a Layer 3 (routed) interface or *routed VLAN interface* (*RVI*) need to be filtered, use a router firewall filter. Apply the filter to the interface at the `[edit family inet]` hierarchy level. You can also apply a router firewall filter on a loopback interface.

Before you choose the interface or VLAN on which to apply a firewall filter, understand how that placement can affect traffic flow to other interfaces. In general, apply a filter close to the source device if the filter matches on source or destination IP addresses, IP protocols, or protocol information—such as ICMP message types, and TCP or UDP port numbers. However, you should apply a filter close to the destination device if the filter matches *only* on a source IP address. When you apply a filter too close to the source device, the filter could prevent that source device from accessing other services that are available on the network.

> **NOTE**: Egress firewall filters do not affect the flow of locally generated control packets from the Routing Engine.

6. In which direction should the firewall filter be applied?

   You typically configure different actions for traffic entering an interface than you configure for traffic exiting an interface.

7. How many filters should I create?

   See "Planning the Number of Firewall Filters to Create" on page 1721 for information about how many firewall filters you can apply.

RELATED DOCUMENTATION

Overview of Policers | **2198**

Understanding How Firewall Filters Are Evaluated | **854**

Configuring Firewall Filters | **1826**

Planning the Number of Firewall Filters to Create

IN THIS SECTION

- How to Increase the Number of Firewall Filters | 1721
- TCAM | 1722
- Avoid Configuring too Many Filters | 1723
- Configuring TCAM Error Messages | 1723
- How to Increase the Scale of Firewall Filters Using Profiles | 1723
- How Policers can Limit Egress Filters | 1729
- Planning for Filter-Specific Policers | 1730
- Planning for Filter-Based Forwarding | 1730

How to Increase the Number of Firewall Filters

You can increase the number of firewall filters on your device several ways:

- (QFX5220) To create more than 512 egress VLAN filters, specify the first VLAN ID as 6, the second VLAN ID as 7, the third VLAN ID as 8 and so forth. For each VLAN that you configure, the number increases by 1 and continues through VLAN ID 1029. If you want to create fewer than 512 egress VLAN filters, but want the total number of terms in those filters to be more than 512, make sure that you number your VLAN IDs this same way. Otherwise, the total number of allowed terms or filters will be less than 1024 and will remain at 512.

- Starting in Junos OS Release 19.1R1, you can increase the number of egress VLAN firewall filters on the QFX5110 from 1024 to 2048 by using the `egress-to-ingress` option. You include this option under the `from` statement at the `[edit firewall]` hierarchy.

  Starting in Junos OS Evolved Release 19.4R2, you can configure up to 2000 egress firewall filters on the QFX5220 by including the `egress-scale` option under the `eracl-profile` statement at the `[edit system packet-forwarding-option firewall]` hierarchy level. This feature is supported only in the egress direction (routed traffic exiting the device).

  Consider this following when configuring this feature:

  - You cannot apply filters with the same match conditions to different egress VLANs or Layer 3 interfaces.

  - You cannot apply egress scaling on GRE interfaces.

- If a packet matches multiple filters with different qualifiers and you apply on different egress interfaces, this can lead to unpredictable behavior.
- You can only configure the `egress-scale` option in global mode. The new cli configuration will be provided in global mode. Once a user configures ERACL group in egress-scale (egress to ingress) mode, he will not be able to configure ERACL the older way i.e., without using IFP tcam space. In other words, ERACL in mixed mode will not be supported.

TCAM

Ternary content addressable memory (TCAM) for firewall filters is divided into slices that accommodate 256 terms. When you configure a firewall filter, all terms in a memory slice must be in filters of the same type and applied in the same direction. A memory slice is reserved as soon as you commit a filter. For example, if you create a port filter and apply it in the input direction, a memory slice is reserved that only stores ingress port filters. If you create and apply only one ingress port filter and that filter has only one term, the rest of this slice is unused and is unavailable for other filter types.

NOTE: In an EVPN environment, QFX5200 Series switches support up to 512 TCAM entries.

For example, let's say that you create and apply 256 ingress port filters with one term each so that one memory slice is filled. This leaves two more memory slices available for ingress filters. (In this case, the maximum number of ingress terms is 768.) If you then create and apply an ingress Layer 3 filter with one term, another memory slice is reserved for ingress Layer 3 filters. As before, the rest of the slice is unused and is unavailable for different filter types. Now there is one memory slice available for any ingress filter type.

Now assume that you create and apply a VLAN ingress filter. The final memory slice is reserved for VLAN ingress filters. Memory allocation for ingress filters (once again assuming one term per filter) is:

- Slice 1: Filled with 256 ingress port filters. You cannot commit any more ingress port filters.
- Slice 2: Contains one ingress Layer 3 filter with one term. You can commit 255 more terms in ingress Layer 3 filters.
- Slice 3: Contains one ingress VLAN filter with one term. You can commit 255 more terms in ingress VLAN filters.

Here is another example. Assume that you create 257 ingress port filters with one term per filter–that is, you create one more term than a single memory slice can accommodate. When you apply the filters and commit the configuration, the filter memory allocation is:

- Slice 1: Filled with 256 ingress port filters. You cannot apply any more ingress port filters.
- Slice 2: Contains one ingress port filter. You can apply 255 more terms in ingress port filters.

- Slice 3: This slice is unassigned. You can create and apply 256 terms in ingress filters of any type (port, Layer 3, or VLAN), but all the filters must be of the same type.

NOTE: All of the above examples also apply to egress filters. The difference is that four memory slices are used because IPv4 and IPv6 Layer 3 filters are stored in separate slices. The memory slices for egress filters are the same size as those for ingress filters, so the maximum number of filters will be the same (1024).

Avoid Configuring too Many Filters

If you violate any of these restrictions and commit a configuration that is not in compliance, Junos OS rejects the excessive filters. For example, if you configure 300 ingress port filters and 300 ingress Layer 3 filters and try to commit the configuration, Junos OS does the following (again assuming one term per filter):

- Accepts the 300 ingress port filters (storing them in two memory slices).
- Accepts the first 256 ingress Layer 3 filters it processes (storing them in the third memory slice).
- Rejects the remaining 44 ingress Layer 3 filters.

NOTE: Make sure that you delete the excessive filters (for example, the remaining 44 ingress Layer 3 filters) from the configuration before you reboot the device. If you reboot a device that has a noncompliant configuration, it's hard to predict which filters were installed after the reboot. Using the example above, the 44 ingress Layer 3 filters that were originally rejected might be installed, and 44 of the port filters that were originally accepted might be rejected.

Configuring TCAM Error Messages

If you lack TCAM space and are unable to install a firewall filter, you can configure your switch to send error messages the following ways:

- Enter `set system syslog file filename pfe emergency` to send error messages to a syslog file.
- Enter `set system syslog console pfe emergency` to send error messages to the console.
- Enter `set system syslog user user-login pfe emergency` to send error messages to an SSH terminal session.

How to Increase the Scale of Firewall Filters Using Profiles

When you configure a firewall filter, the term statement in the firewall filter configuration provides an extensive set of match conditions. Match conditions are the fields and values that a packet must contain to be considered a match. You can define a single or multiple match conditions based on your

requirement. When a packet matches a filter, the device takes the action specified in the term. The scalability of firewall filters generally depends on the number of match conditions used.

In typical deployment scenarios, you will only need to use a subset of match conditions. With the introduction of profiles, you can use one of the available firewall filter profiles with pre-defined match conditions to increase the number of firewall filters used to achieve maximum scale.

You can configure firewall filters profiles for family inet and Ethernet-based switching. Use the profiles configuration statement at the [edit system packet-forwarding-options firewall] hierarchy level to configure firewall filter profiles.

> **NOTE**: When you make changes to the firewall filter profiles, either by selecting a profile, or moving from one profile to another, the packet forwarding engine is restarted, which causes disruption to the traffic flow.

The following table describes the firewall filter profiles and the pre-defined match conditions for inet and Ethernet-based switching.

Table 94: Firewall Filter Profile and Match Conditions

Family Type	Firewall Filter Profiles	Match Condition (pre-defined)	Configuration Hierarchy
inet (IPv4/IPv6)	`profile1`	`ip-source-address` ip-source-prefix-list protocol next-header source-port destination-port first-fragment is-fragment icmp-code icmp-type tcp-established tcp-initial tcp-flags	`[edit system packet-forwarding-options firewall profiles inet profile1]`

Table 94: Firewall Filter Profile and Match Conditions *(Continued)*

Family Type	Firewall Filter Profiles	Match Condition (pre-defined)	Configuration Hierarchy
	profile2	ip-source-address ip6-source-address ip-source-prefix-list ip6-source-prefix-list protocol next-header source-port destination-port first-fragment is-fragment icmp-code icmp-type tcp-established tcp-initial tcp-flags dscp precedence traffic-class ttl hop-limit	`[edit system packet-forwarding-options firewall profiles inet profile2]`
Ethernet Switching	profile1	source-mac-address destination-mac-address	`[edit system packet-forwarding-options firewall profiles ethernet-switching profile1]`

Table 94: Firewall Filter Profile and Match Conditions *(Continued)*

Family Type	Firewall Filter Profiles	Match Condition (pre-defined)	Configuration Hierarchy
	profile2	source-mac-address destination-mac-address ether-type ip-source-address ip-source-prefix-list ip-protocol source-port destination-port next-header	`[edit system packet-forwarding-options firewall profiles ethernet-switching profile2]`

NOTE: When you select a firewall filter profile, you must apply a match condition that is part of the pre-defined match condition subset. If you apply a match condition that is not part of the pre-defined match condition subset of the firewall filter profile, then a commit error occurs. For example, if you select `profile1` for inet filter and apply the match condition as `ip-destination-address`, which is not part of the pre-defined match condition, then you will see an error during the commit operation stating that the `ip-destination-address` match is not part of `profile1` for inet filter.

You can use the `show pfe filter hw profile-info` CLI command to view the details of the firewall filter profiles.

NOTE: Firewall module supports two different profiles (profile-one and profile-two) only on ACX EVO platforms. By default, pfe will come up with profile-two and users will be given a CLI option to allow them to switch to the other profile. Once profile is toggled via cli, pfe will be restarted.

IPV6 IFF filter: Profile-one supports sip6 upto 128 bit . Profile-two supports sip6 upto 64bit.

Ipv6 lo0 filter: Profile-one supports dip6 upto 128bit. Profile-two supports dip6 upto 64bit.

Ipv6-Bgp-flow-spec filters and IPv6-FTF filters will be supported only in profile-two.

Category	PMF Features	Profile Two	Profile One
Family Any	IPv6 IFL Filter	Yes	Yes
	IPv6 IFL Filter - Log/Syslog/Reject Action	No	No
Family Filter	IPv6 IFF Filter	Yes	Yes
	IPv6 IFF Filter - SIP6=128bit match	No	Yes
	IPv6 IFF Filter - L4 Matches	Yes	Yes
	IPv6 IFF Filter - Log/Syslog/Reject Action	Yes	Yes
Lo0 Filter	IPv6 Lo0 Filter - Hw Support	Yes	Yes
	IPv6 Lo0 Filter - VRF Attachment	Yes	No
	IPv6 Lo0 Filter - Sw Support	Yes	Yes
	dip6 128bit match	No	Yes

To achieve maximum firewall filter scale, it is recommended to apply interface level (Layer 2 or Layer 3) filters and distribute the filters equally across the interfaces of different packet processing pipelines. Each set of interfaces is mapped to a packet processing pipeline which handles the packets received on

those interfaces. In this case, the firewall filters are installed to the packet processing pipeline's TCAM memory space mapped to the respective interface.

When a packet enters an interface, the firewall filter performs filtering actions on the packet in the packet processing pipeline based on the match conditions before exiting an egress interface. In the case of multiple packet processing pipelines, when packets enter a device through multiple interfaces, the firewall filter performs filtering actions on the packets passing through the respective packet processing pipelines. Having the interface level filters distributed equally across the interfaces of different packet processing pipelines gives better scale

You can use the `show pfe filter hw port-pipe-info` CLI command to view the details of the packet processing pipeline that each physical interface is mapped to. The output of this CLI command also provides information about the firewall filters installed on a packet processing pipeline. You can use this information to plan and distribute firewall filters across pipelines to achieve maximum scale.

The following sample output of the `show pfe filter hw port-pipe-info` CLI command show the details of the packet processing pipeline that each physical interface is mapped to:

```
user@host> show pfe filter hw port-pipe-info
IFD        Pipe
et-0/0/0    1
et-0/0/1    1
...
et-0/0/10    0
```

How Policers can Limit Egress Filters

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some more examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.

- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

You can stop this problem from happening by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

Planning for Filter-Specific Policers

You can configure policers to be filter-specific. This means that Junos OS creates only one policer instance no matter how many times the policer is referenced. When you do this, rate limiting is applied in aggregate, so if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, the total bandwidth allowed by the filter is 1 Gbps. However, the behavior of a filter-specific policer is affected by how the firewall filter terms that reference the policer are stored in ternary content addressable memory (TCAM). If you create a filter-specific policer and reference it in multiple firewall filter terms, the policer allows more traffic than expected if the terms are stored in different TCAM slices. For example, if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms that are stored in three separate memory slices, the total bandwidth allowed by the filter is 3 Gbps, not 1 Gbps.

To prevent this unexpected behavior from happening, use the information about TCAM slices above to organize your configuration file so that all the firewall filter terms that reference a given filter-specific policer are stored in the same TCAM slice.

Planning for Filter-Based Forwarding

You can use firewall filters along with virtual routing instances to specify different routes for packets to travel in their networks. To set up this feature–called filter-based forwarding, you specify a filter and match criteria and then specify the virtual routing instance to send packets to. Filters used in this way also consume memory in an additional TCAM. See *Understanding FIP Snooping, FBF, and MVR Filter*

Scalability for more information. The section *FBF Filter VFP TCAM Consumption* in this topic specifically addresses the number of supported filters when using filter-based forwarding.

NOTE: Filter-based forwarding does not work with IPv6 interfaces on some Juniper switches.

Release History Table

Release	Description
19.4R2-EVO	Starting in Junos OS Evolved Release 19.4R2, you can configure up to 2000 egress firewall filters on the QFX5220 by including the `egress-scale` option under the `eracl-profile` statement at the `[edit system packet-forwarding-option firewall]` hierarchy level.
19.1R1	Starting in Junos OS Release 19.1R1, you can increase the number of egress VLAN firewall filters on the QFX5110 from 1024 to 2048 by using the `egress-to-ingress` option.

RELATED DOCUMENTATION

- Understanding How Firewall Filters Are Evaluated | **854**
- Understanding Firewall Filter Planning | **1719**
- Configuring Firewall Filters | **1826**
- Understanding Filter-Based Forwarding | **1854**

Firewall Filter Match Conditions and Actions (QFX and EX Series Switches)

IN THIS SECTION

- Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700) | **1732**
- Firewall Filter Match Conditions and Actions (QFX5220 and the QFX5130-32CD) | **1762**

Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700)

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the fields and values that a packet must contain to be considered a match. You can define single or multiple match conditions in *match statements*. You can also include no match statement, in which case the term matches all packets.

When a packet matches a filter, a switch takes the action specified in the term. In addition, you can specify action modifiers to count, mirror, rate-limit, and classify packets. If no match conditions are specified for the term, the switch accepts the packet by default.

- Table 96 on page 1734 describes the match conditions you can specify when configuring a firewall filter. Some of the numeric range and bit-field match conditions allow you to specify a text synonym. To see a list of all the synonyms for a match condition, type `?` at the appropriate place in a statement.
- Table 97 on page 1758 shows the actions that you can specify in a term.
- Table 98 on page 1759 shows the action modifiers you can use to count, mirror, rate-limit, and classify packets.

For match conditions on specific switches, these limitations apply:

Table 95: Limitations

(QFX5100, QFX5110, QFX5200) When using filter-based forwarding on IPv6 interfaces, only these match conditions are supported in the (ingress direction): `source-address`, `destination-address`, `source-prefix-list`, `destination-prefix-list`, `source-port`, `destination-port`, `hop-limit`, `icmp-type`, and `next-header`.
(QFX5110) When you enable the `egress-to-ingress` option under the `[edit firewall]` hierarchy, only `accept`, `discard`, and `count` actions are supported.
(QFX5100, QFX5110, QFX5120, QFX5130-32CD, QFX5220, QFX5700) In an EVPN-VXLAN environment, only these match conditions are supported: `source-address`, `destination-address`, `source-port`, `destination-port`, `ttl`, `ip-protocol`, and `user-vlan-id`.
(QFX5100, QFX5110, QFX5200) You cannot apply a firewall filter in the egress direction on a EVPN-VXLAN IRB interface.
(QFX5700) You cannot apply a firewall filter in the egress direction on a loopback interface.

(QFX5100, QFX5110) If you are using firewall filters to implement MAC filtering in an EVPN-VXLAN environment, see *MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment* for the supported match conditions.

(QFX5100, QFX5110) For each firewall filter that you apply to a VXLAN, you can specify `family ethernet-switching` to filter Layer 2 (Ethernet) packets, or `family inet` to filter on IRB interfaces. You cannot apply a firewall filter in the egress direction on IRB interfaces.

On switches that do not support Layer 2 features, use only those match conditions that are valid for IPv4 and IPv6 interfaces.

(QFX5120, EX4650) Starting with Junos Release 21.4R1, the following match conditions are supported in an EVPN-VXLAN environment on QFX5120, and EX4650: `gbp-src-tag`, and `gbp-dst-tag`.

Starting in Junos OS Release 21.4R1, the source-port-range-optimize and the destination-port-range-optimize conditions are supported under `[edit firewall family ethernet-switching filter <filter-name> term <term-name> from]` hierarchy level. This considerably reduces the TCAM space usage. In QFX5100 switches with source-port-range-optimize and destination-port-range-optimize match conditions configured, upto 24 non-contiguous source-port range and destination-port range match conditions are supported. If more than 24 non-contiguous match conditions are configured, it might throw an error.

Starting with Junos Release 22.4R1, the following match conditions are supported for GBP tagging in an EVPN-VXLAN environment on supported EX4100, EX4400, EX4650, and QFX5120 Series switches: `ip-version ipv4`, `ip-version ipv6`, `mac-address`, `vlan-id`, `interface` + `vlan-id` combination, and `interface`.

Starting with Junos Release 23.2R1, new IPV4 and IPv6 L4 matches are supported for policy enforcement on the EX4100 series, EX4400 series, EX4650 series, QFX5120-32C and QFX5120-48Y switches.

Starting in Junos OS Release 23.4R1 and later, the `vlan-id` *vlan list* | *vlan-range* and `interface` *interface-list* match conditions are supported for GBP tagging in an EVPN-VXLAN environment on supported EX4100, EX4400, EX4650, and QFX5120 Series switches. The EX4100 switches do not support VLAN and PORT+VLAN based GBP.

Table 96: Supported Match Conditions for Firewall Filters

Match Condition	Description	Direction and Interface
`arp-type`	ARP request packet or ARP reply packet.	Egress and ingress interfaces.
`destination-address` *ip-address*	IP destination address field, which is the address of the final destination node.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.
`destination-mac-address` *mac-address*	Destination media access control (MAC) address of the packet.	Ingress ports, VLANs and IPv4 (inet) interfaces. Egress ports and VLANs.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`destination-port` *value*	TCP or UDP destination port field. Typically, you specify this match in conjunction with the `protocol` match statement. For the following well-known ports you can specify 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)`, `nntp (119)`, `ntalk (518)`, `ntp (123)`, `pop3 (110)`, `pptp (1723)`, `printer (515)`, `radacct (1813)`,`radius (1812)`, `rip (520)`, `rkinit (2108)`,	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
	`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)`, `xdmcp (177)`, `zephyr-clt (2103)`, `zephyr-hm (2104)`	
`destination-port range-optimize` *range*	Match a range of TCP or UDP port ranges while using the available memory more efficiently. Using this condition allows you to configure more firewall filters than if you configure individual destination ports. (Not supported with filter-based forwarding.)	Ingress ports, VLANs, IPv4 (inet) interfaces.
`destination-prefix-list` *prefix-list*	IP destination prefix list field. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces and IPv6 (inet6) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`dscp` *`value`*	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. You can specify DSCP in hexadecimal, binary, or decimal form. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `be`—best effort (default) • `ef (46)`—as defined in RFC 3246, *An Expedited Forwarding PHB*. • `af11 (10)`, `af12 (12)`, `af13 (14)`; `af21 (18)`, `af22 (20)`, `af23 (22)`; `af31 (26)`, `af32 (28)`, `af33 (30)`; `af41 (34)`, `af42 (36)`, `af43 (38)` These four classes, with three drop precedences in each class, for a total of 12 code points, are defined in RFC 2597, *Assured Forwarding PHB*. • `cs0`, `cs1`, `cs2`, `cs3`, `cs4`, `cs5`, `cs6`, `cs7`, `cs5`	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`ether-type` *value*	Ethernet type field of a packet. The EtherType value specifies what protocol is being transported in the Ethernet frame. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `aarp (0x80F3)`—EtherType value AARP • `appletalk (0x809B)`—EtherType value AppleTalk • `arp (0x0806)`—EtherType value ARP • `fcoe (0x8906)`—EtherType value FCoE • `fip (0x8914)`—EtherType value FIP • `ipv4 (0x0800)`—EtherType value IPv4 • `ipv6 (0x08DD)`—EtherType value IPv6 • `mpls-multicast (0x8848)`—EtherType value MPLS multicast • `mpls-unicast (0x8847)`—EtherType value MPLS unicast • `oam (0x88A8)`—EtherType value OAM • `ppp (0x880B)`—EtherType value PPP	Ingress ports and VLANs. Egress ports and VLANs.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
	• `pppoe-discovery (0x8863)`—EtherType value PPPoE Discovery Stage • `pppoe-session (0x8864)`—EtherType value PPPoE Session Stage • `sna (0x80D5)`—EtherType value SNA	
`egress-to-ingress`	Include this option to increase the number of egress VLAN firewall filter terms from 1024 to 2048.	Egress VLAN IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`exp`	Match on MPLS EXP bits.	Ingress MPLS interfaces. Egress MPLS interfaces.
`fragment-flags` *`value`*	IP fragmentation flags. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): • `is-fragment` • `dont-fragment (0x4000)` • `more-fragments (0x2000)` • `reserved (0x8000)`	Ingress ports and VLANs.
`gbp-dst-tag`	Match the destination tag, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN.	Not applicable

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`gbp-src-tag`	Match the source tag, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN.	Not applicable

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`icmp-code` *value*	ICMP code field. Because the meaning of the value depends upon the associated `icmp-type`, you must specify a value for `icmp-type` along with a value for `icmp-code`. 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: • *IPv4:* parameter-problem—`ip-header-bad (0)`, `required-option-missing (1)` • *IPv6:* parameter-problem—`ip6-header-bad (0)`, `unrecognized-next-header (1)`, `unrecognized-option (2)` • `redirect`—`redirect-for-network (0)`, `redirect-for-host (1)`, `redirect-for-tos-and-net (2)`, `redirect-for-tos-and-host (3)` • `time-exceeded`—`ttl-eq-zero-during-reassembly (1)`, `ttl-eq-zero-during-transit (0)` • *IPv4:* unreachable—`network-unreachable (0)`, `host-unreachable (1)`, `protocol-unreachable (2)`, `port-unreachable (3)`, `fragmentation-needed (4)`, `source-route-failed (5)`, `destination-network-unknown (6)`, `destination-host-unknown (7)`, `source-host-isolated (8)`, `destination-`	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
	`network-prohibited (9)`, `destination-host-prohibited (10)`, `network-unreachable-for-TOS (11)`, `host-unreachable-for-TOS (12)`, `communication-prohibited-by-filtering (13)`, `host-precedence-violation (14)`, `precedence-cutoff-in-effect (15)` • *IPv6:* unreachable—`address-unreachable (3)`, `administratively-prohibited (1)`, `no-route-to-destination (0)`, `port-unreachable (4)`	
`hop-limit` *value*	Match the specified hop limit or set of hop limits. Specify a single value or a range of values from 0 through 255.	Ingress and egress IPv6 (inet6) interfaces. **NOTE**: Not supported in the egress direction on the QFX3500, QFX3600, QFX5100, QFX5120, QFX5110, QFX5200, and QFX5210 switches.
`ip-version ipv4` *<ip address>* | *<prefix-list>* `ip-version ipv6` *<ip address>* | *<prefix-list>*	Match the IPv4 or IPv6 source or destination address, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress and egress (system wide).
`ip-version ipv4 destination-port` *DST_PORT*	Match the TCP/UDP destination port, for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`ip-version ipv4 source-port` *SRC_PORT*	Match the TCP/UDP source port, for use with for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 ip-protocol` *PROTOCOL*	Match the IP protocol type, for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 is-fragment`	Match if the packet is a fragment, for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 fragment-flag` *FLAGS*	Match the fragment flags (in symbolic or hex formats), for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 ttl` *Value*	IP Time-to-live (TTL) field in decimal. The value can be 1-255. For use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`ip-version ipv4 tcp-flags`*FLAGS*	Match one or more TCP flags (in symbolic or hex formats), for use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 tcp-initial`	Match the first TCP packet of a connection. For use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv4 tcp-established`	Match the packets of an established TCP connection, for use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv6 source-port` *SRC_PORT*	Match the TCP/UDP source port, for use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv6 destination-port` *DST_PORT*	Match the TCP/UDP destination port, for use with for use with GBP policy filter L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`ip-version ipv6 next-header` *PROTOCOL*	Match the next header protocol type, for use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv6 tcp-flags` *FLAGS*	Match the TCP flags, for use with GBP policy L4 matches, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv6 tcp-initial`	Match the initial packets of an established TCP connection, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.
`ip-version ipv6 tcp-established`	Match the packets of an established TCP connection, as described in: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN	Ingress only.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`icmp-type` *value*	ICMP message type field. Typically, you specify this match in conjunction with the `protocol` match statement to determine which protocol is being used on the port. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): *IPv4:* `echo-reply (0)`, `destination unreachable (3)`, `source-quench (4)`, `redirect (5)`, `echo-request (8)`, `IPv4 (inet)-advertisement (9)`, `IPv4 (inet)-solicit (10)`, `time-exceeded (11)`, `parameter-problem (12)`, `timestamp (13)`, `timestamp-reply (14)`, `info-request (15)`, `info-reply (16)`, `mask-request (17)`, `mask-reply (18)` *IPv6:* `destination-unreachable (1)`, `packet-too-big (2)`, `time-exceeded (3)`, `parameter-problem (4)`, `echo-request (128)`, `echo-reply (129)`, `membership-query (130)`, `membership-report (131)`, `membership-termination (132)`, `router-solicit (133)`, `router-advertisement (134)`, `neighbor-solicit (135)`, `neighbor-advertisement (136)`, `redirect (137)`, `router-renumbering (138)`, `node-information-request (139)`, `node-information-reply (140)` See also `icmp-code` *variable*.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`interface` *interface-name* | *<interface_list>*	Interface on which the packet is received, including the logical unit. You can include the wildcard character (*) as part of an interface name or logical unit. **NOTE**: An interface from which a packet is sent cannot be used as a match condition. Match a list of interfaces under the same term in a filter. For use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`ip-destination-address` *address*	IPv4 address that is the final destination node address for the packet.	Ingress ports and VLANs.
`ip6-destination-address` *address*	IPv6 address that is the final destination node address for the packet.	Ingress ports and VLANs. (You cannot simultaneously apply a filter with this match criterion to a Layer 2 port and VLAN that includes that port.)
`ip-options`	Specify `any` to create a match if anything is specified in the options field in the IP header.	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`ip-precedence` *ip-precedence-field*	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).	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.
`ip-protocol` *number*	IP protocol field.	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.
`ip-source-address` *address*	IPv4 address of the source node sending the packet.	Ingress ports and VLANs.
`ip6-source-address` *address*	IPv6 address of the source node sending the packet.	Ingress ports and VLANs. (You cannot simultaneously apply a filter with this match criterion to a Layer 2 port and VLAN that includes that port.)
`ip-version` *address*	IP version of the packet. Use this condition to match IPv4 or IPv6 header fields in traffic that arrives on a Layer 2 port or VLAN interface.	Ingress ports and VLANs.
`is-fragment`	Using this condition causes a match if the More Fragments flag is enabled in the IP header or if the fragment offset is not zero.	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`l2-encap-type llc-non-snap`	Match on logical link control (LLC) layer packets for non-Subnet Access Protocol (SNAP) Ethernet Encapsulation type.	Ingress ports and VLANs. Egress ports and VLANs.
`label`	Match on MPLS label bits.	Ingress MPLS interfaces. Egress MPLS interfaces.
`learn-vlan-id` *number*	Matches the ID of a normal VLAN or the ID of the outer (service) VLAN (for Q-in-Q VLANs). The acceptable values are 1-4095. **NOTE**: Not supported on QFX3600, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5220, EX4600, EX4650, EX4400, EX4100 and EX4300-MP switches. Use the `user-vlan-id` match condition to match the outer VLAN ID.	Ingress ports and VLANs. Egress ports and VLANs.
`mac-address` *mac-address*	Match the source media access control (MAC) address, for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN.	Ingress and egress (system wide) .

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`next-header`	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6 (58)`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	Ingress ports, VLANs, and IPv6 (inet6) interfaces. Egress IPv6 (inet6) interfaces.
`packet-length`	Packet length in bytes. You must enter a value between 0 and 65535.	Ingress ports, VLANs, IPv4 (inet), and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`payload-protocol`	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6 (58)`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)` **NOTE**: Not supported on the QFX3500, QFX3600, QFX5100, QFX5110, QFX5200, QFX5210 switches.	Ingress ports, VLANs, and IPv6 (inet6) interfaces. Egress IPv6 (inet6) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`Port qualifier`	The port qualifier will install two entries in the packet forwarding engine. One with the source-port and second one with the destination-port.	Ingress ports, VLANs, IPv4 (inet), and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`precedence` *value*	IP precedence bits in the type-of-service (ToS) byte in the IP header. (This byte can also used for the DiffServ DSCP.) In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): • `routine (0)` • `priority (1)` • `immediate (2)` • `flash (3)` • `flash-override (4)` • `critical-ecp (5)` • `internet-control (6)` • `net-control (7)`	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`protocol` *type*	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	Ingress ports, VLANs and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`rat-type` *`tech-type-value`*	Match the radio-access technology (RAT) type specified in the 8-bit Tech-Type field of Proxy Mobile IPv4 (PMIPv4) access technology type extension. The technology type specifies the access technology through which the mobile device is connected to the access network. Specify a single value, a range of values, or a set of values. You can specify a technology type as a numeric value from 0 through 255 or as a system keyword. • Numeric value 1 matches IEEE 802.3. • Numeric value 2 matches IEEE 802.11a/b/g. • Numeric value 3 matches IEEE 802.16e • Numeric value 4 matches IEEE 802.16m. • Text string `eutran` matches 4G. • Text string `geran` matches 2G. • Text string `utran` matches 3G.	Egress and ingress IPv4 (inet) interfaces.
`sample`	Sample the packet traffic. Apply this option only if you have enabled traffic sampling.	Egress and ingress IPv4 (inet) interfaces.
`source-address` *`ip-address`*	IP source address field, which is the address of the node that sent the packet.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`source-mac-address` *mac-address*	Source media access control (MAC) address of the packet.	Ingress ports and VLANs. Egress ports and VLANs.
`source-port` *value*	TCP or UDP source port. Typically, you specify this match in conjunction with the `protocol` match statement. In place of the numeric field, you can specify one of the text synonyms listed under `destination-port`.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`source-port range-optimize` *range*	Match a range of TCP or UDP port ranges while using the available memory more efficiently. Using this condition allows you to configure more firewall filters than if you configure individual source ports. (Not supported with filter-based forwarding.)	Ingress ports, VLANs, IPv4 (inet) interfaces.
`source-prefix-list` *prefix-list*	IP source prefix list. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`tcp-established`	Matches packets of an established TCP three-way handshake connection (SYN, SYN-ACK, ACK). The only packet not matched is the first packet of the handshake since only the SYN bit is set. For this packet, you must specify `tcp-initial` as the match condition. When you specify `tcp-established`, the switch does not implicitly verify that the protocol is TCP. You must also specify the `protocol tcp` match condition.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`tcp-flags` *value*	One or more TCP flags: • `ack (0x10)` • `fin (0x01)` • `push (0x08)` • `rst (0x04)` • `syn (0x02)` • `urgent (0x20)`	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`tcp-initial`	Match the first TCP packet of a connection. A match occurs when the TCP flag `SYN` is set and the TCP flag `ACK` is not set. When you specify `tcp-initial`, a switch does not implicitly verify that the protocol is TCP. You must also specify the `protocol tcp` match condition.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`traffic-class`	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. This field was previously used as the type-of-service (ToS) field in IPv4, and, the semantics of this field (for example, DSCP) are identical to those of IPv4. You can specify one of the following text synonyms (the field values are also listed): `af11 (10)`, `af12 (12)`, `af13 (14)`, `af21 (18)`, `af22 (20)`, `af23 (22)`, `af31 (26)`, `af32 (28)`, `af33 (30)`, `af41 (34)`, `af42 (36)`, `af43 (38)`, `cs0 (0)`, `cs1 (8)`, `cs2 (16)`, `cs3 (24)`, `cs4 (32)`, `cs5 (40)`, `cs6 (48)`, `cs7 (56)`, `ef (46)`	Ingress ports, VLANs, and IPv6 (inet6) interfaces. Egress IPv6 (inet6) interfaces.
`ttl` *value*	IP Time-to-live (TTL) field in decimal. The value can be 1-255.	Ingress IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.
`user-vlan-1p-priority` *value*	Matches the specified 802.1p VLAN priority in the range `0-7`.	Ingress and egress ports and VLANs.

Table 96: Supported Match Conditions for Firewall Filters *(Continued)*

Match Condition	Description	Direction and Interface
`user-vlan-id` *number*	Matches the ID of the inner (customer) VLAN for a Q-in-Q VLAN. The acceptable values are 1-4095. **NOTE**: For QFX3600, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4600, EX4650, EX4400, EX4100 and EX4300-MP switches, use `user-vlan-id` to match the ID of the outer VLAN. For QFX5220 Series switches, and MX and ACX Series routers, use *learn*-`vlan-id` to match the ID of the outer VLAN, and `user-vlan-id` to match the ID of the inner VLAN. Previously, you could use `user-vlan-id` to match the outer VLAN ID.	Ingress and egress ports and VLANs.
`vlan-id` *<vlan id> | <vlan-range> | <vlan list>*	Match the VLAN identifier, *vlan-range* (the first and last VLAN ID number for the group of VLANs), or *vlan list* (list of numbers) for use with micro-segmentation on a VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN. **NOTE**: Not supported on the EX4100 switches.	Ingress and egress (system wide)

Use `then` statements to define actions that should occur if a packet matches all conditions in a `from` statement. Table 97 on page 1758shows the actions that you can specify in a term. (If you do not include a `then` statement, the system accepts packets that match the filter.)

Table 97: Actions for Firewall Filters

Action	Description
`accept`	Accept a packet. This is the default action for packets that match a term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.
`reject` *`message-type`*	Discard a packet and send a “destination unreachable” ICMPv4 message (type 3). To log rejected packets, configure the `syslog` action modifier. You can specify one of the following message types: `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`, the system sends a TCP reset if the packet is a TCP packet; otherwise nothing is sent. If you do not specify a message type, the ICMP notification “destination unreachable” is sent with the default message “communication administratively filtered.” **NOTE**: The `reject` action is supported on ingress interfaces only.
`routing-instance` *`instance-name`*	Forward matched packets to a virtual routing instance.
`vlan` *`VLAN-name`*	Forward matched packets to a specific VLAN. **NOTE**: The `vlan` action is supported on ingress interfaces only. **NOTE**: This action is not supported on OCX series switches.

You can also specify the action modifiers listed in Table 98 on page 1759 to count, mirror, rate-limit, and classify packets.

Table 98: Action Modifiers for Firewall Filters

Action Modifier	Description
`analyzer` *analyzer-name*	(Non-ELS platforms) Mirror traffic (copy packets) to an analyzer configured at the `[edit ethernet-switching-options analyzer]` hierarchy level. You can specify port mirroring for ingress port, VLAN, and IPv4 (inet) firewall filters only.
`count` *counter-name*	Count the number of packets that match the term.
`decapsulate [gre |` *routing-instance*`]`	De-encapsulate GRE packets or forward de-encapsulated GRE packets to the specified routing instance
`dscp` *value*	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. You can specify DSCP in hexadecimal, binary, or decimal form. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `be`—best effort (default) • `ef (46)`—as defined in RFC 3246, *An Expedited Forwarding PHB*. • `af11 (10)`, `af12 (12)`, `af13 (14)`; `af21 (18)`, `af22 (20)`, `af23 (22)`; `af31 (26)`, `af32 (28)`, `af33 (30)`; `af41 (34)`, `af42 (36)`, `af43 (38)` These four classes, with three drop precedences in each class, for a total of 12 code points, are defined in RFC 2597, *Assured Forwarding PHB*. • `cs0`, `cs1`, `cs2`, `cs3`, `cs4`, `cs5`, `cs6`, `cs7`, `cs5`

Table 98: Action Modifiers for Firewall Filters *(Continued)*

Action Modifier	Description
`forwarding-class` *class*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `mcast` • `network-control` • `no-loss` **NOTE**: To configure a forwarding class, you must also configure loss priority.
`gbp-src-tag` (QFX5120 and EX4650 only)	Set the group based policy source tag (0..65535) for use with micro-segmentation on VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN.
`gbp-tag` (EX4100, EX4400, EX4650 and QFX5120)	Set the group based policy source tag (1..65535) for use with micro-segmentation on VXLAN, as described here: Example: Micro and Macro Segmentation using Group Based Policy in a VXLAN. **NOTE**: Applies to Junos OS releases 22.4R1 and later.
`interface`	Switch the traffic to the specified interface without performing a lookup on it. This action is valid only when the filter is applied on ingress.
`log`	Log the packet's header information in the Routing Engine. To view this information, enter the `show firewall log` operational mode command. **NOTE**: The `log` action modifier is supported on ingress interfaces only.

Table 98: Action Modifiers for Firewall Filters *(Continued)*

Action Modifier	Description
`loss-priority (low | medium-low | medium-high | high)`	Set the packet loss priority (PLP). **NOTE**: The `loss-priority` action modifier is supported on ingress interfaces only. **NOTE**: The `loss-priority` action modifier is not supported in combination with the `policer` action.
`policer` *`policer-name`*	Send packets to a policer (for the purpose of applying rate limiting). You can specify a policer for ingress port, VLAN, IPv4 (inet), IPv6 (inet6), and MPLS filters. **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.
`port-mirror`	(ELS platforms) Mirror traffic (copy packets) to an output interface configured in a port-mirroring instance at the `[edit forwarding-options port-mirroring]` hierarchy level. You can specify port mirroring for ingress port, VLAN, and IPv4 (inet) firewall filters only.
`port-mirror-instance` *`port-mirror-instance-name`*	(ELS platforms) Mirror traffic to a port-mirroring instance configured at the `[edit forwarding-options port-mirroring]` hierarchy level. You can specify port mirroring for ingress port, VLAN, and IPv4 (inet) firewall filters only. **NOTE**: This action modifier is not supported on OCX series switches.
`syslog`	Log an alert for this packet. **NOTE**: The `syslog` action modifier is supported on ingress interfaces only.

Table 98: Action Modifiers for Firewall Filters *(Continued)*

Action Modifier	Description
`three-color-policer` *three-color-policer-name*	Send packets to a three-color policer (for the purpose of applying rate limiting). You can specify a three-color policer for ingress and egress port, VLAN, IPv4 (inet), IPv6 (inet6), and MPLS filters. **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.

SEE ALSO

Firewall Filter Flexible Match Conditions | 862

Firewall Filter Match Conditions and Actions (QFX5220 and the QFX5130-32CD)

This topic describes the supported firewall filter match conditions, actions, and action modifiers for the QFX5220-CD, QFX5220-128C, and QFX5130-32CD switches.

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the fields and values that a packet must contain to be considered a match. You can define single or multiple match conditions in *match statements*. You can also include no match statement, in which case the term matches all packets.

When a packet matches a filter, a switch takes the action specified in the term. If you apply no match condition, the switch accepts the packet by default.

- Table 99 on page 1763 shows the match conditions for IPv4 (`inet`) and the IPv6 (`inet6`) interfaces. It also contains the match conditions for ports and VLANs (`ethernet-switching`).
- Table 100 on page 1777 shows the actions and the action modifiers that you can specify in a term.

NOTE: For match conditions, some of the numeric range and the bit-field match conditions allow you to specify a text synonym. To see a list of all the synonyms for a match condition, type `?` at the appropriate place in a statement.

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches)

Match Condition	Description	Direction and Interface
`arp-type`	ARP request packet or an ARP reply packet.	Ingress and egress ports and VLANs
`destination-address` *ip-address*	IP destination address field, which is the address of the final destination node.	Ingress and egress IPv4 and IPv6 interfaces Ingress ports and VLANs
`destination-mac-address` *mac-address*	Destination MAC address of the packet.	Ingress and egress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`destination-port` *value*	TCP or UDP destination port field. You must specify this match with the `protocol` match statement for IPv4 traffic, or the `next-header` match statement for IPv6 traffic. For the following well-known ports and port numbers you can specify text synonyms. `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)`, `nntp (119)`, `ntalk (518)`, `ntp (123)`, `pop3 (110)`, `pptp (1723)`, `printer (515)`, `radacct (1813)`,`radius (1812)`, `rip (520)`, `rkinit (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)`,	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces. Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
	`xdmcp (177)`, `zephyr-clt (2103)`, `zephyr-hm (2104)`	
`destination-port range-optimize` *range*	Match a range of the TCP or UDP port ranges while using the available memory more efficiently. Using this condition allows you to configure more firewall filters than if you configure individual destination ports. (Not supported with filter-based forwarding.)	Ingress IPv4 interfaces
`destination-prefix-list` *prefix-list*	IP destination prefix list field. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress and egress IPv4 and IPv6 interfaces Ingress ports and VLANs.

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`dscp` *`value`*	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. You can specify DSCP in hexadecimal, binary, or decimal form. In place of the numeric value, you can specify one of the following text synonyms and field listed. • `be`—best effort (default) • `ef (46)`—as defined in RFC 3246, *An Expedited Forwarding PHB*. • `af11 (10)`, `af12 (12)`, `af13 (14)`; `af21 (18)`, `af22 (20)`, `af23 (22)`; `af31 (26)`, `af32 (28)`, `af33 (30)`; `af41 (34)`, `af42 (36)`, `af43 (38)` These four classes, with three drop precedences in each class, for a total of 12 code points, are defined in RFC 2597, *Assured Forwarding PHB*. • `cs0`, `cs1`, `cs2`, `cs3`, `cs4`, `cs5`, `cs6`, `cs7`, `cs5`	Ingress and egress IPv4 interfaces Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`ether-type` *`value`*	Ethernet type field of a packet. The EtherType value specifies what protocol is being transported in the Ethernet frame. In place of the numeric value, you can specify one of the following text synonyms. The field values are also listed. • `aarp (0x80F3)`—EtherType value AARP • `appletalk (0x809B)`—EtherType value AppleTalk • `arp (0x0806)`—EtherType value ARP • `fcoe (0x8906)`—EtherType value FCoE • `fip (0x8914)`—EtherType value FIP • `ipv4 (0x0800)`—EtherType value IPv4 • `ipv6 (0x08DD)`—EtherType value IPv6 • `mpls-multicast (0x8848)`—EtherType value MPLS multicast • `mpls-unicast (0x8847)`—EtherType value MPLS unicast • `oam (0x88A8)`—EtherType value OAM • `ppp (0x880B)`—EtherType value PPP • `pppoe-discovery (0x8863)`—EtherType value PPPoE Discovery Stage • `pppoe-session (0x8864)`—EtherType value PPPoE Session Stage • `sna (0x80D5)`—EtherType value SNA	Ingress and egress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
first-fragment	Match if the packet is the first fragment of a fragmented packet. Avoiding matching the packet if it is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of 0. This match condition is an alias for the bit-field match condition fragment-offset 0 match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: `first-fragment` and `is-fragment`.	Ingress IPv4 interfaces

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`icmp-code` *`value`*	ICMP code field. Because the meaning of the value depends upon the associated `icmp-type`, you must specify a value for `icmp-type` along with a value for `icmp-code`. 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: • *IPv4:* parameter-problem—`ip-header-bad (0)`, `required-option-missing (1)` • *IPv6:* parameter-problem—`ip6-header-bad (0)`, `unrecognized-next-header (1)`, `unrecognized-option (2)` • `redirect—redirect-for-network (0)`, `redirect-for-host (1)`, `redirect-for-tos-and-net (2)`, `redirect-for-tos-and-host (3)` • `time-exceeded—ttl-eq-zero- during-reassembly (1)`, `ttl-eq-zero-during-transit (0)` • *IPv4:* unreachable—`network-unreachable (0)`, `host-unreachable (1)`, `protocol-unreachable (2)`, `port-unreachable (3)`, `fragmentation-needed (4)`, `source-route-failed (5)`, `destination-network-unknown (6)`, `destination-host-unknown (7)`, `source-host-isolated (8)`, `destination-network-prohibited (9)`, `destination-host-prohibited (10)`, `network-unreachable-for-TOS (11)`, `host-unreachable-for-TOS (12)`, `communication-prohibited-by-filtering (13)`, `host-precedence-violation (14)`, `precedence-cutoff-in-effect (15)` • *IPv6:* unreachable—`address-unreachable (3)`, `administratively-prohibited (1)`, `no-route-to-destination (0)`, `port-unreachable (4)`	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`icmp-type` *value*	ICMP message type field. You must specify this match along with the `protocol` match statement. This match determines which protocol is being used on the port for IPv4 traffic, or the `next-header` match statement for IPv6 traffic. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): *IPv4:* `echo-reply (0)`, `destination unreachable (3)`, `source-quench (4)`, `redirect (5)`, `echo-request (8)`, `IPv4 (inet)-advertisement (9)`, `IPv4 (inet)-solicit (10)`, `time-exceeded (11)`, `parameter-problem (12)`, `timestamp (13)`, `timestamp-reply (14)`, `info-request (15)`, `info-reply (16)`, `mask-request (17)`, `mask-reply (18)` *IPv6:* `destination-unreachable (1)`, `packet-too-big (2)`, `time-exceeded (3)`, `parameter-problem (4)`, `echo-request (128)`, `echo-reply (129)`, `membership-query (130)`, `membership-report (131)`, `membership-termination (132)`, `router-solicit (133)`, `router-advertisement (134)`, `neighbor-solicit (135)`, `neighbor-advertisement (136)`, `redirect (137)`, `router-renumbering (138)`, `node-information-request (139)`, `node-information-reply (140)` See also `icmp-code` *variable*.	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs
`interface` *interface-name*	Interface on which the packet is received, including the logical unit. You can include the wildcard character (*) as part of an interface name or logical unit. **NOTE**: An interface from which a packet is sent cannot be used as a match condition.	Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`ip-destination-address` *address*	IPv4 address that is the final destination node address for the packet.	Ingress ports and VLANs
`ip-options`	Specify `any` to create a match if anything is specified in the options field in the IP header.	Ingress IPv4 interfaces
`ip-protocol` *number*	IP protocol field.	Ingress ports and VLANs
`ip-precedence` *ip-precedence-field*	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).	Ingress ports and VLANs
`ip-source-address` *address*	IPv4 address of the source node sending the packet.	Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`ip-version` *address*	IP version of the packet. Use this condition to match IPv4 or IPv6 header fields in traffic that arrives on a Layer 2 port or VLAN interface.	Ingress ports and VLANs
`is-fragment`	Using this condition causes a match if the More Fragments flag is enabled in the IP header or if the fragment offset is not zero.	Ingress and egress IPv4 interfaces (QFX5220) Ingress IPv4 interfaces (QFX5130)
`learn-vlan-id` *number*	VLAN identifier for MAC learning.	Ingress and egress ports and VLANs (QFX5220) Ingress ports and VLANS (QFX5130)
`learn-vlan-1p-priority` *value*	Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from 0 through 7.	Ingress ports and VLANs
`next-header`	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6 (58)`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	Ingress and egress IPv6 interfaces

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`packet-length`	Packet length in bytes. You must enter a value between 0 and 65535.	Ingress IPv4 and IPv6 interfaces
`precedence` *value*	IP precedence bits in the type-of-service (ToS) byte in the IP header. (This byte can also used for the DiffServ DSCP.) In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): • `routine (0)` • `priority (1)` • `immediate (2)` • `flash (3)` • `flash-override (4)` • `critical-ecp (5)` • `internet-control (6)` • `net-control (7)`	Ingress and egress IPv4 interfaces
`protocol` *type*	IP protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`tcp (4)	Ingress and egress IPv4 interfaces. Ingress IPv4 interfaces and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`source-address` *ip-address*	IP source address field, which is the address of the node that sent the packet.	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs
`source-mac-address` *mac-address*	Source media access control (MAC) address of the packet.	Ingress and egress IPv4 interfaces and VLANs
`source-port` *value*	TCP or UDP source port. You must specify this match in conjunction with the `protocol` match statement for IPv4 traffic, or the `next-header` match statement for IPv6 traffic. In place of the numeric field, you can specify one of the text synonyms listed under `destination-port`.	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs
`source-port range-optimize` *range*	Match a range of TCP or UDP port ranges while using the available memory more efficiently. Using this condition allows you to configure more firewall filters than if you configure individual source ports. (Not supported with filter-based forwarding.)	Ingress IPv4 interfaces
`source-prefix-list` *prefix-list*	IP source prefix list. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`tcp-established`	Match TCP packets of an established TCP session (packets other than the first packet of a connection). This is an alias for `tcp-flags "(ack | rst)"`. This match condition does not implicitly check that the protocol is TCP. To check this, specify the `protocol tcp` match condition.	Ingress and egress IPv4 interfaces (QFX5220) Ingress and egress IPv4 interfaces (QFX5130) Ingress IPv6 interfaces (QFX5130)
`tcp-flags` *value*	TCP flags (only one value is supported): • `ack (0x10)` • `fin (0x01)` • `push (0x08)` • `rst (0x04)` • `syn (0x02)` • `urgent (0x20)`	Ingress and egress IPv4 interfaces Ingress IPv6 interfaces Ingress ports and VLANs
`tcp-initial`	Match the first TCP packet of a connection. A match occurs when the TCP flag `SYN` is set and the TCP flag `ACK` is not set. When you specify `tcp-initial`, a switch does not implicitly verify that the protocol is TCP. You must also specify the `protocol tcp` match condition. See `protocol` *type*.	Ingress and egress IPv4 interfaces (QFX5220) Ingress and egress IPv4 interfaces, Ingress IPv6 interfaces (QFX5130)

Table 99: Supported Match Conditions (QFX5220 and QFX5130-32CD Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`traffic-class`	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. This field was previously used as the type-of-service (ToS) field in IPv4, and, the semantics of this field (for example, DSCP) are identical to those of IPv4. You can specify one of the following text synonyms (the field values are also listed): `af11 (10)`, `af12 (12)`, `af13 (14)`, `af21 (18)`, `af22 (20)`, `af23 (22)`, `af31 (26)`, `af32 (28)`, `af33 (30)`, `af41 (34)`, `af42 (36)`, `af43 (38)`, `cs0 (0)`, `cs1 (8)`, `cs2 (16)`, `cs3 (24)`, `cs4 (32)`, `cs5 (40)`, `cs6 (48)`, `cs7 (56)`, `ef (46)`	Ingress and egress IPv6 interfaces
`ttl` *`value`*	IP Time-to-live (TTL) field in decimal. The value can be 1-255.	Ingress and egress IPv4 interfaces
`user-vlan-id` *`number`*	Matches the ID of the inner (customer) VLAN for a Q-in-Q VLAN. The acceptable values are 1-4095.	Ingress ports and VLANs (QFX5130)
`user-vlan-1p-priority` *`value`*	Matches the specified 802.1p VLAN priority in the range `0-7`.	Ingress ports and VLANs (QFX5130)

Use `then` statements to define actions that should occur if a packet matches all conditions in a `from` statement. Table 100 on page 1777 shows the actions that you can specify in a term. (If you do not include a `then` statement, the system accepts packets that match the filter.)

NOTE: For egress IPv4 interfaces, IPv6 interfaces, and egress ports, you can only apply the accept, discard, and count actions. For egress VLANs, you can only apply the accept action.

Table 100: Actions and Action Modifiers

Action	Description
`accept`	Accept a packet. This is the default action for packets that match a term.
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.
`count` *`counter-name`*	Count the number of packets that match the term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.
`forwarding-class` *`class`*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `mcast` • `network-control` • `no-loss` **NOTE**: To configure a forwarding class, you must also configure loss priority.
`log`	Log the packet's header information in the Routing Engine. To view this information, enter the `show firewall log` operational mode command.

Table 100: Actions and Action Modifiers *(Continued)*

Action	Description
`loss-priority (low | medium-low | medium-high | high)`	Set the packet loss priority (PLP). **NOTE**: The `loss-priority` action modifier is supported on ingress IPv4 interfaces only. **NOTE**: The `loss-priority` action modifier is not supported in combination with the `policer` action.
`policer` *`policer-name`*	Send packets to a policer (for the purpose of applying rate limiting). **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.
`port-mirror`	Mirror traffic (copy packets) to an output interface configured in a port-mirroring instance at the `[edit forwarding-options port-mirroring]` hierarchy level.
`port-mirror-instance` *`port-mirror-instance-name`*	Mirror traffic to a port-mirroring instance configured at the `[edit forwarding-options port-mirroring]` hierarchy level. You can specify port mirroring for ingress port, VLAN, and IPv4 (inet) firewall filters only.

Table 100: Actions and Action Modifiers *(Continued)*

Action	Description
`reject` *`message-type`*	Discard a packet and send a “destination unreachable” ICMPv4 message (type 3). To log rejected packets, configure the `syslog` action modifier. You can specify one of the following message types: `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.` If you do not specify a message type, the ICMP notification “destination unreachable” is sent with the default message “communication administratively filtered.” **NOTE**: The `reject` action is supported on ingress IPv4 interfaces only.
`three-color-policer` *`three-color-policer-name`*	Send packets to a three-color policer (for the purpose of applying rate limiting). **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action. **NOTE**: The `color-aware` and `color-blind` policers are not supported. By default, traffic is treated as `color-blind`.
`vlan` *`VLAN-name`*	Forward matched packets to a specific VLAN. **NOTE**: The `vlan` action is only supported on ingress ports and VLANs. This action is not supported on QFX5130 switches.

SEE ALSO

- Overview of Firewall Filters (OCX Series) | **846**
- Understanding How Firewall Filters Are Evaluated | **854**
- Configuring Firewall Filters | **1826**

Overview of Policers | 2198

Firewall Filter Match Conditions and Actions (QFX10000 Switches)

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the fields and values that a packet must contain to be considered a match. You can define single or multiple match conditions in *match statements*. You can also include no match statement, in which case the term matches all packets.

When a packet matches a filter, the switch takes the action specified in the term. In addition, you can specify action modifiers to count, mirror, rate-limit, and classify packets. If no match conditions are specified for the term, the switch accepts the packet by default.

This topic describes the various match conditions, actions, and action modifiers that you can define in firewall filters on QFX10000 switches. For similar information about other QFX switches, see "Firewall Filter Match Conditions and Actions (QFX and EX Series Switches)" on page 1731.

- Table 101 on page 1780 describes the match conditions you can specify when configuring a firewall filter. Some of the numeric range and bit-field match conditions allow you to specify a text synonym. To see a list of all the synonyms for a match condition, type ? at the appropriate place in a statement.
- Table 102 on page 1794 shows the actions that you can specify in a term.
- Table 103 on page 1795 shows the action modifiers you can use to count, mirror, rate-limit, and classify packets.

Table 101: Supported Match Conditions (QFX10000 Switches)

Match Condition	Description	Direction and Interface
`destination-address` *`ip-address`*	IP destination address field, which is the address of the final destination node.	Ingress IPv4 (inet) interfaces and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Ingress IRB interface for EVPN/VXLAN fabric, where applicable
`destination-mac-address` *`mac-address`*	Destination media access control (MAC) address of the packet.	Ingress ports and VLANs. Egress ports and VLANs.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`destination-port` *value*	TCP or UDP destination port field. Typically, you specify this match in conjunction with the `protocol` match statement. For the following well-known ports you can specify 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)`, `nntp (119)`, `ntalk (518)`, `ntp (123)`, `pop3 (110)`, `pptp (1723)`, `printer (515)`, `radacct (1813)`,`radius (1812)`, `rip (520)`, `rkinit (2108)`, `smtp (25)`, `snmp (161)`, `snmptrap (162)`, `snpp (444)`, `socks (1080)`, `ssh (22)`, `sunrpc (111)`, `syslog (514)`,	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. ingress IRB interface for EVPN/VXLAN fabric, where applicable

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
	`tacacs-ds (65)`, `talk (517)`, `telnet (23)`, `tftp (69)`, `timed (525)`, `who (513)`, `xdmcp (177)`, `zephyr-clt (2103)`, `zephyr-hm (2104)`	
`destination-prefix-list` *`prefix-list`*	IP destination prefix list field. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`dscp` *value*	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. You can specify DSCP in hexadecimal, binary, or decimal form. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `be`—best effort (default) • `ef (46)`—as defined in RFC 3246, *An Expedited Forwarding PHB*. • `af11 (10)`, `af12 (12)`, `af13 (14)`; `af21 (18)`, `af22 (20)`, `af23 (22)`; `af31 (26)`, `af32 (28)`, `af33 (30)`; `af41 (34)`, `af42 (36)`, `af43 (38)` These four classes, with three drop precedences in each class, for a total of 12 code points, are defined in RFC 2597, *Assured Forwarding PHB*. • `cs0`, `cs1`, `cs2`, `cs3`, `cs4`, `cs5`, `cs6`, `cs7`, `cs5`	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress ports, VLANs, and IPv4 (inet) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`ether-type` *value*	Ethernet type field of a packet. The EtherType value specifies what protocol is being transported in the Ethernet frame. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `aarp (0x80F3)`—EtherType value AARP • `appletalk (0x809B)`—EtherType value AppleTalk • `arp (0x0806)`—EtherType value ARP • `fcoe (0x8906)`—EtherType value FCoE • `fip (0x8914)`—EtherType value FIP • `ipv4 (0x0800)`—EtherType value IPv4 • `ipv6 (0x08DD)`—EtherType value IPv6 • `mpls-multicast (0x8848)`—EtherType value MPLS multicast • `mpls-unicast (0x8847)`—EtherType value MPLS unicast • `oam (0x88A8)`—EtherType value OAM • `ppp (0x880B)`—EtherType value PPP • `pppoe-discovery (0x8863)`—EtherType value PPPoE Discovery Stage • `pppoe-session (0x8864)`—EtherType value PPPoE Session Stage • `sna (0x80D5)`—EtherType value SNA	Ingress ports and VLANs. Egress ports and VLANs.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`forwarding-class` *class*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `network-control` • `no-loss`	Egress IPv4 (inet) and IPv6 (inet6) interfaces.
`fragment-flags` *value*	IP fragmentation flags. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): • `is-fragment` • `dont-fragment (0x4000)` • `more-fragments (0x2000)` • `reserved (0x8000)`	Ingress ports, VLANs, and IPv4 (inet) interfaces.
`hop-limit` *value*	Match the specified hop limit or set of hop limits. Specify a single value or a range of values from 0 through 255.	Ingress and egress IPv6 (inet6) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`icmp-code` *value*	ICMP code field. Because the meaning of the value depends upon the associated `icmp-type`, you must specify a value for `icmp-type` along with a value for `icmp-code`. 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: • *IPv4:* parameter-problem—`ip-header-bad (0)`, `required-option-missing (1)` • *IPv6:* parameter-problem—`ip6-header-bad (0)`, `unrecognized-next-header (1)`, `unrecognized-option (2)` • `redirect—redirect-for-network (0)`, `redirect-for-host (1)`, `redirect-for-tos-and-net (2)`, `redirect-for-tos-and-host (3)` • `time-exceeded—ttl-eq-zero- during-reassembly (1)`, `ttl-eq-zero-during-transit (0)` • *IPv4:* unreachable—`network-unreachable (0)`, `host-unreachable (1)`, `protocol-unreachable (2)`, `port-unreachable (3)`, `fragmentation-needed (4)`, `source-route-failed (5)`, `destination-network-unknown (6)`, `destination-host-unknown (7)`, `source-host-isolated (8)`, `destination-network-prohibited (9)`, `destination-host-prohibited (10)`, `network-unreachable-for-TOS (11)`, `host-unreachable-for-TOS (12)`, `communication-prohibited-by-filtering (13)`, `host-`	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
	precedence-violation (14), precedence-cutoff-in-effect (15) • *IPv6:* unreachable—address-unreachable (3), administratively-prohibited (1), no-route-to-destination (0), port-unreachable (4)	
icmp-type *value*	ICMP message type field. Typically, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): *IPv4:* echo-reply (0), destination unreachable (3), source-quench (4), redirect (5), echo-request (8), IPv4 (inet)-advertisement (9), IPv4 (inet)-solicit (10), time-exceeded (11), parameter-problem (12), timestamp (13), timestamp-reply (14), info-request (15), info-reply (16), mask-request (17), mask-reply (18) *IPv6:* destination-unreachable (1), packet-too-big (2), time-exceeded (3), parameter-problem (4), echo-request (128), echo-reply (129), membership-query (130), membership-report (131), membership-termination (132), router-solicit (133), router-advertisement (134), neighbor-solicit (135), neighbor-advertisement (136), redirect (137), router-renumbering (138), node-information-request (139), node-information-reply (140) See also icmp-code *variable*.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`interface` *interface-name*	Interface on which the packet is received, including the logical unit. You can include the wildcard character (*) as part of an interface name or logical unit. **NOTE**: An interface from which a packet is sent cannot be used as a match condition.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`ip-destination-address` *address*	IPv4 address that is the final destination node address for the packet.	Ingress ports, egress ports, and VLANs. Ingress IRB interface for EVPN/VXLAN fabric, where applicable
`ip-options`	Specify `any` to create a match if anything is specified in the options field in the IP header.	Ingress ports, VLANs, and IPv4 (inet) interfaces.
`ip-precedence` *ip-precedence-field*	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).	Ingress ports and VLANs. Egress ports and VLANs.
`ip-protocol` *number*	IP protocol field.	Ingress ports and VLANs. Egress ports and VLANs. Ingress IRB interface for EVPN/VXLAN fabric, where applicable
`ip-source-address` *address*	IPv4 address of the source node sending the packet.	Ingress ports and VLANs. Egress ports and VLANs. Ingress IRB interface for EVPN/VXLAN fabric, where applicable

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`ip-version` *address*	IP version of the packet. Use this condition to match IPv4 or IPv6 header fields in traffic that arrives on a Layer 2 port or VLAN interface.	Ingress ports and VLANs. Egress ports and VLANs.
`is-fragment`	Using this condition causes a match if the More Fragments flag is enabled in the IP header or if the fragment offset is not zero.	Ingress ports, VLANs, and IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.
`learn-1p-priority` *number*	Matches the specified IEEE 802.1p VLAN priority bits in the range `0-7`.	Ingress ports and VLANs. Egress ports and VLANs.
`learn-vlan-id` *number*	Matches the ID of a normal VLAN or the ID of the outer (service) VLAN (for Q-in-Q VLANs). To use filter memory most efficiently and maximize the number of possible filters, use this condition in addition to `user-id` when you want to match on the inner (customer) VLAN ID. The acceptable values are 1-4095.	Ingress ports and VLANs. Egress ports and VLANs. Ingress IRB interface for EVPN/VXLAN fabric, where applicable
`loss-priority (low | medium-low | medium-high | high)`	Set the packet loss priority (PLP). **NOTE**: The `loss-priority` action modifier is not supported in combination with the `policer` action.	Egress IPv4 (inet) and IPv6 (inet6) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`next-header` *value*	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6 (58)`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	Ingress IPv6 (inet6) interfaces. Egress IPv6 (inet6) interfaces.
`packet-length` *number*	Packet length in bytes. You must enter a number between 0 and 65535.	Ingress ports, VLANs, IPv4 (inet), and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`precedence` *value*	IP precedence bits in the type-of-service (ToS) byte in the IP header. (This byte can also used for the DiffServ DSCP.) In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): • `routine (0)` • `priority (1)` • `immediate (2)` • `flash (3)` • `flash-override (4)` • `critical-ecp (5)` • `internet-control (6)` • `net-control (7)`	Ingress IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`protocol` *type*	IPv4 or IPv6 protocol value. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	Ingress IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces.
`source-address` *ip-address*	IP source address field, which is the address of the node that sent the packet.	Ingress IPv4 (inet) interfaces and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces and IPv6 (inet6) interfaces. Ingress IRB interface for EVPN/VXLAN fabric, where applicable
`source-mac-address` *mac-address*	Source media access control (MAC) address of the packet.	Ingress ports and VLANs. Egress ports and VLANs.
`source-port` *value*	TCP or UDP source port. Typically, you specify this match in conjunction with the `protocol` match statement. In place of the numeric field, you can specify one of the text synonyms listed under `destination-port`.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Ingress IRB interface for EVPN/VXLAN fabric, where applicable

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`source-prefix-list` *prefix-list*	IP source prefix list. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.
`tcp-established`	Match packets of an established TCP connection. This condition matches packets other than those used to set up a TCP connection—that is, three-way handshake packets are not matched. When you specify `tcp-established`, a switch does not implicitly verify that the protocol is TCP. You must also specify the `protocol tcp` match condition.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`tcp-flags` *value*	One or more TCP flags: • `ack (0x10)` • `fin (0x01)` • `push (0x08)` • `rst (0x04)` • `syn (0x02)` • `urgent (0x20)`	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`tcp-initial`	Match the first TCP packet of a connection. A match occurs when the TCP flag `SYN` is set and the TCP flag `ACK` is not set. When you specify `tcp-initial`, a switch does not implicitly verify that the protocol is TCP. You must also specify the `protocol tcp` match condition.	Ingress ports, VLANs, IPv4 (inet) interfaces, and IPv6 (inet6) interfaces. Egress IPv4 (inet) interfaces.
`traffic-class` *value*	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. This field was previously used as the type-of-service (ToS) field in IPv4, and, the semantics of this field (for example, DSCP) are identical to those of IPv4. You can specify one of the following text synonyms (the field values are also listed): `af11 (10)`, `af12 (12)`, `af13 (14)`, `af21 (18)`, `af22 (20)`, `af23 (22)`, `af31 (26)`, `af32 (28)`, `af33 (30)`, `af41 (34)`, `af42 (36)`, `af43 (38)`, `cs0 (0)`, `cs1 (8)`, `cs2 (16)`, `cs3 (24)`, `cs4 (32)`, `cs5 (40)`, `cs6 (48)`, `cs7 (56)`, `ef (46)`	Ingress IPv6 (inet6) interfaces. Egress IPv6 (inet6) interfaces.
`ttl` *value*	IP Time-to-live (TTL) field in decimal. The value can be 1-255.	Ingress IPv4 (inet) interfaces. Egress IPv4 (inet) interfaces. Ingress IRB interface for EVPN/VXLAN fabric, where applicable

Table 101: Supported Match Conditions (QFX10000 Switches) *(Continued)*

Match Condition	Description	Direction and Interface
`user-vlan-id` *number*	Matches the ID of the inner (customer) VLAN in a Q-in-Q VLAN. To use filter memory most efficiently and maximize the number of possible filters, use in combination with `learn-vlan-id` to match the outer (service) VLAN ID. The acceptable values are 1-4095.	Ingress ports and VLANs. Egress ports and VLANs.

Use `then` statements to define actions that should occur if a packet matches all conditions in a `from` statement. Table 102 on page 1794 shows the actions that you can specify in a term. (If you do not include a `then` statement, the system accepts packets that match the filter.)

Table 102: Actions

Action	Description
`accept`	Accept a packet. This is the default action for packets that match a term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.

Table 102: Actions *(Continued)*

Action	Description
`reject` *message-type*	Discard a packet and send a "destination unreachable" ICMPv4 message (type 3). To log rejected packets, configure the `syslog` action modifier. You can specify one of the following message types: `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`, the system sends a TCP reset if the packet is a TCP packet; otherwise nothing is sent. If you do not specify a message type, the ICMP notification "destination unreachable" is sent with the default message "communication administratively filtered." **NOTE**: The `reject` action is supported on ingress interfaces only.
`routing-instance` *instance-name*	Forward matched packets to a virtual routing instance. (The only supported instance type is `virtual-router`.) Packets can be forwarded to the default instance.
`vlan` *VLAN-name*	Forward matched packets to a specific VLAN. **NOTE**: The `vlan` action is supported on ingress interfaces only. **NOTE**: This action is not supported on OCX series switches.

You can also specify the action modifiers listed in Table 103 on page 1795 to count, mirror, rate-limit, and classify packets.

Table 103: Action Modifiers

Action Modifier	Description
`count` *counter-name*	Count the number of packets that match the term.

Table 103: Action Modifiers *(Continued)*

Action Modifier	Description
`forwarding-class` *class*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `mcast` • `network-control` • `no-loss` **NOTE**: To configure a forwarding class, you must also configure loss priority.
`log`	Log the packet's header information in the Routing Engine. To view this information, enter the `show firewall log` operational mode command. **NOTE**: The `log` action modifier is supported on ingress interfaces only.
`loss-priority (low | medium-low | medium-high | high)`	Set the packet loss priority (PLP). **NOTE**: The `loss-priority` action modifier is supported on ingress interfaces only. **NOTE**: The `loss-priority` action modifier is not supported in combination with the `policer` action.
`policer` *policer-name*	Send packets to a policer (for the purpose of applying rate limiting). You can specify a policer for ingress and egress port, VLAN, IPv4 (inet), and IPv6 (inet6) firewall filters. **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.

Table 103: Action Modifiers *(Continued)*

Action Modifier	Description
`port-mirror`	(ELS platforms) Mirror traffic (copy packets) to an output interface configured in a port-mirroring instance at the `[edit forwarding-options port-mirroring]` hierarchy level. You can specify port mirroring for ingress and egress port, VLAN, IPv4 (inet), and IPv6 (inet6) firewall filters.
`port-mirror-instance` *`port-mirror-instance-name`*	(ELS platforms) Mirror traffic to a port-mirroring instance configured at the `[edit forwarding-options port-mirroring]` hierarchy level. You can specify port mirroring for ingress and egress port, VLAN, IPv4 (inet), and IPv6 (inet6) firewall filters. **NOTE**:
`syslog`	Log an alert for this packet. **NOTE**: The `syslog` action modifier is supported on ingress interfaces only.
`three-color-policer` *`three-color-policer-name`*	Send packets to a three-color policer (for the purpose of applying rate limiting). You can specify a three-color policer for ingress and egress port, VLAN, IPv4 (inet), and IPv6 (inet6) filters. **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.

RELATED DOCUMENTATION

- Overview of Firewall Filters (OCX Series) | **846**
- Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | **1731**
- Configuring Firewall Filters | **1826**
- Overview of Policers | **2198**

Firewall Filter Match Conditions and Actions (PTX Series Routers)

IN THIS SECTION

- Firewall Filter Match Conditions and Actions (PTX Series Routers) | **1798**
- IPv6 Firewall Filter Match Conditions and Actions (PTX10001-20C) | **1815**

Firewall Filter Match Conditions and Actions (PTX Series Routers)

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the fields and values that a packet must contain to be considered a match. You can define single or multiple match conditions in *match statements*. You can also include no match statement, in which case the term matches all packets.

When a packet matches a filter, the router takes the action specified in the term. In addition, you can specify action modifiers to count, mirror, rate-limit, and classify packets. If no match conditions are specified for the term, the router accepts the packet by default.

On the PTX10003, you can apply multiple firewall filters to a single interface as a single input list or output list (`filter input-list and output-list`). In this way, you only manage the configuration for a filtering task in a single firewall filter. This gives you flexibility in large environments when you have a device configured with many interfaces. You can do the same on the PTX10008, but the router only supports applying multiple firewall filters to a single input-list.

- Table 104 on page 1798 describes the match conditions you can specify when configuring a firewall filter. Some of the numeric range and bit-field match conditions allow you to specify a text synonym. To see a list of all the synonyms for a match condition, type `?` at the appropriate place in a statement.
- Table 105 on page 1813 shows the actions and action modifiers that you can specify in a term.

Table 104: Supported Match Conditions

Match Condition	Description	Supported Interfaces
`address` *`address`* `[ except ]`	Match the source or destination address field unless the `except` option is included.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`destination-address` *address* `[ except ]`	Match the destination address field unless the `except` option is included. You cannot specify both `address` and `destination-address` match conditions in the same term.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`destination-port` *number*	Match the UDP or TCP destination port field. You must also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. You cannot specify both the `port` and `destination-port` match conditions in the same term. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).	IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.
`destination-port-except` *number*	Do not match the UDP or TCP destination port field. For details, see the `destination-port` match condition.	IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`destination-prefix-list` *name* `[ except ]`	Match destination prefixes in a list unless the `except` option is included. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.
`dscp` *number*	Match the 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. You can specify DSCP in hexadecimal, binary, or decimal form. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • `be`—best effort (default) • `ef (46)`—as defined in RFC 3246, *An Expedited Forwarding PHB*. • `af11 (10)`, `af12 (12)`, `af13 (14)`; `af21 (18)`, `af22 (20)`, `af23 (22)`; `af31 (26)`, `af32 (28)`, `af33 (30)`; `af41 (34)`, `af42 (36)`, `af43 (38)` These four classes, with three drop precedences in each class, for a total of 12 code points, are defined in RFC 2597, *Assured Forwarding PHB*. • `cs0`, `cs1`, `cs2`, `cs3`, `cs4`, `cs5`, `cs6`, `cs7`, `cs5`	IPv4 (inet) and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`dscp-except` *number*	Do not match on the DSCP number. For more information, see the `dscp` match condition.	IPv4 (inet) and IPv6 (inet6) interfaces.
`first-fragment`	Match if the packet is the first fragment of a fragmented packet. Do not match if the packet is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of `0`. This match condition is an alias for the bit-field match condition `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: `first-fragment` and `is-fragment`.	IPv4 (inet) interfaces.
`forwarding-class` *class*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `network-control` • `no-loss`	IPv4 (inet), IPv6 (inet6), and MPLS interfaces.
`forwarding-class-except` *class*	Do not match the forwarding class of the packet. For details, see the `forwarding-class` match condition.	IPv4 (inet), IPv6 (inet6), and MPLS interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`fragment-flags` *number*	Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): `dont-` (0x4), `more-s` (0x2), or `reserved` (0x8).	IPv4 (inet) interfaces.
`fragment-offset` *value*	Match the 13-bit fragment offset field in the IP header. The value is the offset, in 8-byte units, in the overall datagram message to the data fragment. Specify a numeric value, a range of values, or a set of values. An offset value of `0` indicates the first fragment of a fragmented packet. The `first-fragment` match condition is an alias for the `fragment-offset 0` match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`).	IPv4 (inet) interfaces.
`fragment-offset-except` *number*	Do not match the 13-bit fragment offset field.	IPv4 (inet) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`icmp-code` *`message-code`*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. If you configure this match condition, you must also configure the `icmp-type` *`message-type`* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • 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-`	IPv4 (inet) and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
	`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)	
`icmp-code-except` *`message-code`*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.	IPv4 (inet) and IPv6 (inet6) interfaces.
`icmp-type` *`number`*	Match the ICMP message type field. You must also configure `icmp` or `icmpv6` as `protocol` *`next-header`* match type in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `echo-reply` (0), `echo-request` (8), `info-reply` (16), `info-request` (15), `mask-request` (17), `mask-reply` (18), `parameter-problem` (12), `redirect` (5), `router-advertisement` (9), `router-solicit` (10), `source-quench` (4), `time-exceeded` (11), `timestamp` (13), `timestamp-reply` (14), or `unreachable` (3). See also `icmp-code` *`variable`*.	IPv4 (inet) and IPv6 (inet6) interfaces.
`icmp-type-except` *`message-type`*	Do not match the ICMP message type field. For details, see the `icmp-type` match condition.	IPv4 (inet) and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces

`interface` *interface-name*	For ingress filters, match the interface on which the packet was received.
	
	For egress filters, match the interface on which the packet was sent.
	
	NOTE: PTX5000 series routers do not support attaching the `em0.0` interface (the internal link between the routing and packet forwarding engines) to `lo0` (the loopback interface), for example to filter self-originating traffic such as Telnet and SSH by creating a firewall filter on lo0 to match traffic on em0.0. The following code snippet provides context:
	
	`firewall`
	`  filter core-protect {`
	`  term Telnet {`
	`  from {`
	`  protocol tcp;`
	`  destination-port telnet;`
	`  interface em0.0;`
	`  }`
	`  then accept;`
	`  }`
	`  }`
	`}`
		IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.

`interface-except` *number*	Do not match the logical interface on which the packet was received. For details, see the `interface` match condition.	IPv4 (inet) interfaces, and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`is-fragment`	Match if the packet is a trailing fragment of a fragmented packet. Do not match the first fragment of a fragmented packet. **NOTE**: To match both first and trailing fragments, you can use two terms that specify different match conditions (`first-fragment` and `is-fragment`). For PTX10003 routers running Junos OS Evolved, all fragmented packets including the first fragment of fragmented packets will match on any firewall filter term containing an "is-fragment" match.	IPv4 (inet) interfaces.
`loss-priority` *level*	Match the packet loss priority (PLP). Specify a single level or multiple levels: `low`, `medium-low`, `medium-high`, or `high`. **NOTE**: The `loss-priority` action modifier is not supported in combination with the `policer` action.	IPv4 (inet), IPv6 (inet6), and MPLS interfaces.
`loss-priority-except` *level*	Do not match the PLP level. For details, see the `loss-priority` match condition.	IPv4 (inet), IPv6 (inet6), and MPLS interfaces.
`next-header` *header-type*	Match the first 8-bit next header field in the packet. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstops` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `mobility` (135), `no-next-header` (59), `ospf` (89), `pim` (103), `routing` (43), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).	IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`next-header-except` *header-type*	Do not match the 8-bit Next Header field that identifies the type of header between the IPv6 header and payload. For details, see the `next-header` match type.	IPv6 (inet6) interfaces
`packet-length` *bytes*	Match the 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. You can also specify a range of values to be matched.	IPv4 (inet), and IPv6 (inet6) interfaces.
`packet-length-except` *bytes*	Do not match the length of the received packet, in bytes. For details, see the `packet-length` match type.	IPv4 (inet), and IPv6 (inet6) interfaces.
`port` *number*	Match the UDP or TCP source or destination port field. You must also configure the `protocol udp` or `protocol tcp` match statement in the same term to specify which protocol is being used on the port. You cannot configure the `destination-port` match condition or the `source-port` match condition in the same term. In place of the numeric value, you can specify one of the text synonyms listed under `destination-port`.	IPv4 (inet), and IPv6 (inet6) interfaces.
`port-except` *number*	Do not match either the source or destination UDP or TCP port field. For details, see the `port` match condition.	IPv4 (inet), and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`precedence` *ip-precedence-value*	Match the 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.	IPv4 (inet) interfaces.
`precedence-except` *ip-precedence-value*	Do not match the IP precedence field.	IPv4 (inet) interfaces.
`protocol` *number*	Match the IPv4 protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the numeric values are also listed): `hop-by-hop (0)`,`icmp (1)`, `icmp6`, `igmp (2)`, `ipip (4)`, `tcp (6)`, `egp (8)`, `udp (17)`, `ipv6 (41)`, `routing (43)`, `fragment (44)`,`rsvp (46)`, `gre (47)`, `esp (50)`, `ah (51)`, `icmp6 (58)`, `no-next-header (59)`, `dstopts (60)`, `ospf (89)`, `pim (103)`, `vrrp (112)`, `sctp (132)`	IPv4 (inet) interfaces.
`protocol-except` *number*	Do not match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): `ah` (51), `dstopts` (60), `egp` (8), `esp` (50), `fragment` (44), `gre` (47), `hop-by-hop` (0), `icmp` (1), `icmp6` (58), `icmpv6` (58), `igmp` (2), `ipip` (4), `ipv6` (41), `ospf` (89), `pim` (103), `rsvp` (46), `sctp` (132), `tcp` (6), `udp` (17), or `vrrp` (112).	IPv4 (inet) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`source-address` *ip-address*	IP source address field, which is the address of the node that sent the packet.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`source-address` *address* `[ except ]`	Match the IP address of the source node sending the packet unless the `except` option is included. If the option is included, do not match the IP address of the source node sending the packet. You cannot specify both the `address` and `source-address` match conditions in the same term.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`source-port` *value*	Match the TCP or UDP source port. You must also configure the `protocol udp` or `protocol tcp` match statement in the same term. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`source-port-except` *number*	Do not match the UDP or TCP source port field. For details, see the `source-port` match condition.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.
`source-prefix-list` *prefix-list*	IP source prefix list. You can define a list of IP address prefixes under a prefix-list alias for frequent use. Define this list at the `[edit policy-options]` hierarchy level.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`tcp-flags` *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • `fin` (0x01) • `syn` (0x02) • `rst` (0x04) • `push` (0x08) • `ack` (0x10) • `urgent` (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. If you configure this match condition, we recommend that you also configure the `protocol tcp` match statement in the same term to specify that the TCP protocol is being used on the port. For IPv4 traffic only, this match condition does not implicitly check whether the datagram contains the first fragment of a fragmented packet. To check for this condition for IPv4 traffic only, use the `first-fragment` match condition.	IPv4 (inet) interfaces and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`traffic-class` *value*	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. This field was previously used as the type-of-service (ToS) field in IPv4, and, the semantics of this field (for example, DSCP) are identical to those of IPv4. You can specify one of the following text synonyms (the field values are also listed): `af11 (10)`, `af12 (12)`, `af13 (14)`, `af21 (18)`, `af22 (20)`, `af23 (22)`, `af31 (26)`, `af32 (28)`, `af33 (30)`, `af41 (34)`, `af42 (36)`, `af43 (38)`, `cs0 (0)`, `cs1 (8)`, `cs2 (16)`, `cs3 (24)`, `cs4 (32)`, `cs5 (40)`, `cs6 (48)`, `cs7 (56)`, `ef (46)`	IPv6 (inet6) interfaces.
`traffic-class-except` *number*	Do not match the 8-bit field that specifies the CoS priority of the packet. For details, see the `traffic-class` match description.	IPv6 (inet6) interfaces.
`ttl` *number*	Match the IPv4 or IPv6 time-to-live number. Specify a TTL value or a range of TTL values. For *number*, you can specify one or more values from `0` through `255`.	IPv4 (inet) and IPv6 (inet6) interfaces.
`ttl-except` *number*	Do not match on the IPv4 or IPv6 TTL number. For details, see the `ttl` match condition.	IPv4 (inet) and IPv6 (inet6) interfaces.

Table 104: Supported Match Conditions *(Continued)*

Match Condition	Description	Supported Interfaces
`vxlan`	Specify a numeric value or range of numeric values for the VNI. Apply the filter on the ingress interface. • `vni` *`vni-value`*—Match the VNI. • `vni-except` *`vni-value`*—Do not match the VNI.	IPv4 (inet) interfaces.

Use `then` statements to define actions that should occur if a packet matches all conditions in a `from` statement. Table 105 on page 1813 shows the actions that you can specify in a term. (If you do not include a `then` statement, the system accepts packets that match the filter.)

Table 105: Actions and Action Modifiers

Action	Description
`accept`	Accept a packet. This is the default action for packets that match a term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.
`count` *`counter-name`*	Count the number of packets that match the term.

Table 105: Actions and Action Modifiers *(Continued)*

Action	Description
`forwarding-class` *class*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `mcast` • `network-control` • `no-loss` **NOTE**: The `forwarding-class` action is supported on IPv4, IPv6, and MPLS interfaces.
`log`	Log the packet's header information in the Routing Engine. To view this information, enter the `show firewall log` operational mode command. **NOTE**: The `log` action modifier is only supported on IPv4 and IPv6 ingress interfaces.
`loss-priority` *level*	Set the packet loss priority (PLP).
`policer` *policer-name*	Send packets to a policer (for the purpose of applying rate limiting). The PTX10003 supports two-color, single-rate three-color (srTCM), and two-rate three-color marker (trTCM) policers. **NOTE**: The `policer` action modifier is not supported in combination with the `loss-priority` action.
`redirect` *instance-name*	(Supported on PTX10004, PTX10008, and PTX10016 devices running Junos Evolved OS Release 22.1R1 only.) Send packets to the P4 controller, as specified in the instance defined at the **[services inline-monitoring instance *instance-name* *controller* p4]** level of the Junos hierarchy.

Table 105: Actions and Action Modifiers *(Continued)*

Action	Description
`reject` *message-type*	Discard a packet and send a "destination unreachable" ICMPv4 or ICMPv6 message (type 3). To log rejected packets, configure the `syslog` action modifier. You can specify one of the following message types: `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, .`  **NOTE**: The `tcp-reset` message type is not supported. If you do not specify a message type, the ICMP notification "destination unreachable" is sent with the default message "communication administratively filtered."
`syslog`	Log an alert for this packet.
`routing-instance` *instance-name*	Forward matched packets to a virtual routing instance. Packets can be forwarded to the default instance. Supported on `virtual-router` and `forwarding instance-types`.

IPv6 Firewall Filter Match Conditions and Actions (PTX10001-20C)

This topic describes the IPv6 firewall filter match conditions, actions, and action modifiers for PTX10001-20C routers.

Each term in a firewall filter consists of *match conditions* and an *action*. Match conditions are the fields and values that a packet must contain to be considered a match. You can define single or multiple match conditions in *match statements*. You can also include the *no match statement*, in which case the term matches all packets.

When a packet matches a filter, the router takes the action specified in the term. You can also specify action modifiers to count, mirror, and classify packets. If no match conditions are specified for the term, the router accepts the packet by default.

NOTE: On PTX10001-20C routers, you can only apply a firewall filter on IPv6 interfaces in the ingress direction.

- Table 106 on page 1816 describes the supported match conditions.
- Table 107 on page 1822 shows the actions that you can specify in a term. If you don't include a `then` statement, the system accepts packets that match the filter.
- Table 108 on page 1822 shows the action modifiers you can use to count, mirror, rate-limit, and classify packets.

Table 106: IPv6 Supported Match Conditions

Match Condition	Description
`address` *address* `[ except ]`	Match the IPv6 source or destination address field unless the `except` option is included. If the option is included, do not match the IPv6 source or destination address field.
`apply-groups`	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
`apply-groups-except`	Specify which groups not to inherit configuration data from. You can specify more than one group name.
`destination-address` *address* `[ except ]`	Match the IPv6 destination address field unless the `except` option is included. If the option is included, do not match the IPv6 destination address field. You cannot specify both the `address` and `destination-address` match conditions in the same term.

Table 106: IPv6 Supported Match Conditions *(Continued)*

Match Condition	Description
`destination-port` *number*	Match the UDP or TCP destination port field. You cannot specify both the `port` and `destination-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. 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), `ldp` (646), `login` (513), `mobileip-agent` (434), `mobilip-mn` (435), `msdp` (639), `netbios-dgm` (138), `netbios-ns` (137), `netbios-ssn` (139), `nfsd` (2049), `nntp` (119), `ntalk` (518), `ntp` (123), `pop3` (110), `pptp` (1723), `printer` (515), `radacct` (1813), `radius` (1812), `rip` (520), `rkinit` (2108), `smtp` (25), `snmp` (161), `snmptrap` (162), `snpp` (444), `socks` (1080), `ssh` (22), `sunrpc` (111), `syslog` (514), `tacacs` (49), `tacacs-ds` (65), `talk` (517), `telnet` (23), `tftp` (69), `timed` (525), `who` (513), or `xdmcp` (177).
`destination-port-except` *number*	Do not match the UDP or TCP destination port field. For details, see the `destination-port` match condition.
`destination-prefix-list` *prefix-list-name* `[ except ]`	Match the IPv6 destination prefix to the specified list unless the `except` option is included. If the option is included, do not match the IPv6 destination prefix to the specified list. The prefix list is defined at the [`edit policy-options prefix-list` *prefix-list-name*] hierarchy level.

Table 106: IPv6 Supported Match Conditions *(Continued)*

Match Condition	Description
`icmp-code` *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the `next-header icmp` or `next-header icmp6` match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: `ip6-header-bad` (0), `unrecognized-next-header` (1), `unrecognized-option` (2) • time-exceeded: `ttl-eq-zero-during-reassembly` (1), `ttl-eq-zero-during-transit` (0) • destination-unreachable: `administratively-prohibited` (1), `address-unreachable` (3), `no-route-to-destination` (0), `port-unreachable` (4)
`icmp-code-except` *message-code*	Do not match the ICMP message code field. For details, see the `icmp-code` match condition.

Table 106: IPv6 Supported Match Conditions *(Continued)*

Match Condition	Description
message-type	Match the ICMP message type field. You must also configure icmp or next-header icmp6 match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): certificate-path-advertisement (149), certificate-path-solicitation (148), destination-unreachable (1), echo-reply (129), echo-request (128), home-agent-address-discovery-reply (145), home-agent-address-discovery-request (144), inverse-neighbor-discovery-advertisement (142), inverse-neighbor-discovery-solicitation (141), membership-query (130), membership-report (131), membership-termination (132), mobile-prefix-advertisement-reply (147), mobile-prefix-solicitation (146), neighbor-advertisement (136), neighbor-solicit (135), node-information-reply (140), node-information-request (139), packet-too-big (2), parameter-problem (4), private-experimentation-100 (100), private-experimentation-101 (101), private-experimentation-200 (200), private-experimentation-201 (201), redirect (137), router-advertisement (134), router-renumbering (138), router-solicit (133), or time-exceeded (3). For private-experimentation-201 (201), you can also specify a range of values within square brackets.
icmp-type-except *message-type*	Do not match the ICMP message type field. For details, see the icmp-type match condition.
next	Continue to the next term in a filter.
next-header *header-type*	Match the first 8-bit Next Header field in the packet. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ah (51), dstops (60), egp (8), esp (50), fragment (44), gre (47), hop-by-hop (0), icmp (1), icmp6 (58), icmpv6 (58), igmp (2), ipip (4), ipv6 (41), mobility (135), no-next-header (59), ospf (89), pim (103), routing (43), rsvp (46), sctp (132), tcp (6), udp (17), or vrrp (112). **NOTE**: next-header icmp6 and next-header icmpv6 match conditions perform the same function. next-header icmp6 is the preferred option. next-header icmpv6 is hidden in the Junos OS CLI.

Table 106: IPv6 Supported Match Conditions *(Continued)*

Match Condition	Description
`next-header-except` *header-type*	Do not match the 8-bit Next Header field that identifies the type of header between the IPv6 header and payload. For details, see the `next-header` match type.
`port` *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the `destination-port` match condition or the `source-port` match condition in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed under `destination-port`.
`port-except` *number*	Do not match the UDP or TCP source or destination port field. For details, see the `port` match condition.
`port-mirror` *instance-name*	Port-mirror the packet.
`port-mirror-instance` *instance-name*	Port mirror a packet for an instance.
`prefix-list` *prefix-list-name* `[ except ]`	Match the prefixes of the source or destination address fields to the prefixes in the specified list unless the `except` option is included. If the option is included, do not match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the `[edit policy-options prefix-list` *prefix-list-name*`]` hierarchy level.
`sample`	Sample the packet.

Table 106: IPv6 Supported Match Conditions *(Continued)*

Match Condition	Description
`source-address` *address* `[ except ]`	Match the IPv6 address of the source node sending the packet unless the `except` option is included. If the option is included, do not match the IPv6 address of the source node sending the packet. You cannot specify both the `address` and `source-address` match conditions in the same term.
`source-port` *number*	Match the UDP or TCP source port field. You cannot specify the `port` and `source-port` match conditions in the same term. If you configure this match condition, we recommend that you also configure the `next-header udp` or `next-header tcp` match condition in the same term to specify which protocol is being used on the port. **NOTE**: For Junos OS Evolved, you must configure the `next-header udp` or `next-header tcp` match statement in the same term. In place of the numeric value, you can specify one of the text synonyms listed with the `destination-port` *number* match condition.
`source-port-except` *number*	Do not match the UDP or TCP source port field. For details, see the `source-port` match condition.
`source-prefix-list` *name* `[ except ]`	Match the IPv6 address prefix of the packet source field unless the `except` option is included. If the option is included, do not match the IPv6 address prefix of the packet source field. Specify a prefix list name defined at the `[edit policy-options prefix-list` *prefix-list-name*`]` hierarchy level.

NOTE: If you specify an IPv6 address in a match condition (the `address`, `destination-address`, or `source-address` match conditions), use the syntax for text representations described in RFC 4291, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see IPv6 Overview and Supported IPv6 Standards.

Table 107: Actions for IPv6 Firewall Filters

Action	Description
`accept`	Accept a packet. This is the default action for packets that match a term.
`discard`	Discard a packet silently without sending an Internet Control Message Protocol (ICMP) message.
`redirect` *`instance-name`*	(Supported on PTX10004, PTX10008, and PTX10016 devices running Junos Evolved OS Release 22.1R1 only.) Send packets to the P4 controller, as specified in the instance defined at the **[services inline-monitoring instance *instance-name* controller p4]** level of the Junos hierarchy.

Table 108: Action Modifiers for IPv6 Firewall Filters

Action Modifier	Description
`count` *`counter-name`*	Count the number of packets that match the term.
`forwarding-class` *`class`*	Classify the packet in one of the following default forwarding classes, or in a user-defined forwarding class: • `best-effort` • `fcoe` • `mcast` • `network-control` • `no-loss` **NOTE**: To configure a forwarding class, you must also configure loss priority.

Table 108: Action Modifiers for IPv6 Firewall Filters *(Continued)*

Action Modifier	Description
`loss-priority (low | medium-low | medium-high | high)`	Set the packet loss priority (PLP).

RELATED DOCUMENTATION

- Overview of Firewall Filters (QFX Series) | **1716**
- Configuring Firewall Filters | **1826**
- Overview of Policers | **2198**
- Understanding Multiple Firewall Filters Applied as a List | **1338**
- Guidelines for Applying Multiple Firewall Filters as a List | **1342**

Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers

This topic provides a list of firewall and policier features available on PTX Packet Transport Routers and compares them with firewall and policing features on T Series routers.

Firewall Filters

Junos OS firewall and policing software on PTX Series Packet Transport Routers supports IPv4 filters, IPv6 filters, MPLS filters, CCC filters, interface policing, LSP policing, MAC filtering, ARP policing, L2 policing, and other features. Exceptions are noted below.

- PTX Series Packet Transport Routers do not support:
  - Egress Forwarding Table Filters
  - Forwarding Table Filters for MPLS/CCC
  - Family VPLS

- PTX Series Packet Transport Routers do not support nested firewall filters. The `filter` statement at the `[edit firewall family family-name filter filter-name term term-name]` hierarchy level is disabled.
- Because no service PICs are present in PTX Series Packet Transport Routers, service filters are not supported for both IPv4 and IPv6 traffic. The `service-filter` statement at `[edit firewall family (inet | inet6)]` hierarchy level is disabled.
- The PTX Series Packet Transport Routers exclude simple filters. These filters are supported on Gigabit Ethernet intelligent queuing (IQ2) and Enhanced Queuing Dense Port Concentrator (EQ DPC) interfaces only. The `simple-filter` statement at the `[edit firewall family inet)]` hierarchy level is disabled.
- Physical interface filtering is not supported. The `physical-interface-filter` statement at the `[edit firewall family family-name filter filter-name]` hierarchy level is disabled.
- The prefix action feature is not supported on PTX Series Packet Transport Routers. The `prefix-action` statement at `[edit firewall family inet]` hierarchy level is disabled.
- On T Series routers, you can collect a variety of information about traffic passing through the device by setting up one or more accounting profiles that specify some common characteristics of the data. The PTX Series Packet Transport Routers do not support accounting configurations for firewall filters. The `accounting-profile` statement at the `[edit firewall family family-name filter filter-name]` hierarchy level is disabled.
- The `reject` action is not supported on the loopback (`lo0`) interface. If you apply a filter to the `lo0` interface and the filter includes a `reject` action, an error message appears.
- PTX Series Packet Transport Routers do not support aggregated ethernet logical interface match conditions. However, child link interface matching is supported.
- PTX Series Packet Transport Routers displays both counts if two different terms in a filter have the same match condition but they have different counts. T Series routers display one count only.
- PTX Series Packet Transport Routers do not have separate policer instances when a filter is bound to multiple interfaces. Use the `interface-specific` configuration statement to create the configuration.
- On PTX Series Packet Transport Routers, when an ingress interface has CCC encapsulation, packets coming in through the ingress CCC interface will not be processed by the egress filters.
- For CCC encapsulation, the PTX Series Packet Transport Routers append an extra 8 bytes for egress Layer 2 filtering. The T Series routers do not. Therefore, egress counters on PTX Series Packet Transport Routers show an extra eight bytes for each packet which impacts policer accuracy.
- On PTX Series Packet Transport Routers, output for the `show pfe statistics traffic` CLI command includes the packets discarded by DMAC and SMAC filtering. On T Series routers, the command

output does not include these discarded packets because MAC filters are implemented in the PIC and not in the FPC.

- The last-fragment packet that goes through a PTX firewall cannot be matched by the `is-fragment` matching condition. This feature is supported on T Series routers.

  A possible workaround on PTX Series Packet Transport Routers is to configure two separate terms with same the actions: one term contains a match to `is-fragment` and the other term contains a match to `fragment-offset -except 0`.

- On PTX Series Packet Transport Routers, MAC pause frames are generated when packet discards exceed 100 Mbps. This occurs only for frame sizes that are less than 105 bytes.

Traffic Policiers

Junos OS firewall and policing software on PTX Series Packet Transport Routers supports IPv4 filters, IPv6 filters, MPLS filters, CCC filters, interface policing, LSP policing, MAC filtering, ARP policing, L2 policing, and other features. Exceptions are noted below.

- PTX Series Packet Transport Routers support ARP policing. T Series routers do not.
- PTX Series Packet Transport Routers do not support LSP policing.
- PTX Series Packet Transport Routers do not support the `hierarchical-policer` configuration statement. .
- PTX Series Packet Transport Routers do not support the `interface-set` configuration statement. This statement groups a number of interfaces into a single, named interface set.
- PTX Series Packet Transport Routers do not support the following policer types for both normal policers and three-color policers:
  - `logical-bandwidth-policer` — Policer uses logical interface bandwidth.
  - `physical-interface-policer` — Policer is a physical interface policer.
  - `shared-bandwidth-policer` — Share policer bandwidth among bundle links.
- When a policer action and forwarding-class, loss-priority actions are configured within the same rule (a *Multifield Classification*), the PTX Series Packet Transport Routers work differently than T Series routers. As shown below, you can configure two rules in the filter to make the PTX filter behave the same as the T Series filter:

  PTX Series configuration:

```
rule-1 {
    match: {x, y, z}
    action: {forwarding-class, loss-prio, next}
```

```
}
    rule-2 {
        match: {x, y, z}
        action: {policer}
    }
```

T Series configuration:

```
rule-1 {
    match: {x, y, z}
    action: {forwarding-class, loss-prio, policer}
}
```

RELATED DOCUMENTATION

- Routing Policies, Firewall Filters, and Traffic Policers User Guide

Configuring Firewall Filters

IN THIS SECTION

- Configuring a Firewall Filter | **1826**
- Configuring Enhanced Egress Firewall Filters (QFX5110 and QFX5220 Switches) | **1830**
- Applying a Firewall Filter to a Port | **1831**
- Applying a Firewall Filter to a VLAN | **1832**
- Applying a Firewall Filter to a Layer 3 (Routed) Interface | **1832**
- Applying a Firewall Filter to a Layer 2 CCC (QFX10000 Switches) | **1834**

Follow the steps in the following sections to configure and apply a firewall filter on your switch.

Configuring a Firewall Filter

To configure a firewall filter:

1. Configure the family address type, filter name, term name, and at least one match condition—for example, match on packets that contain a specific source address.

```
[edit]
user@switch# set firewall family ethernet-switching filter ingress-port-filter term term-one
from source-address 192.0.2.14
```

   - To filter Layer 2 traffic (port or VLAN), specify the family address type `ethernet-switching`.
   - To filter Layer 3 (routed) traffic, specify the family address type (`inet` for IPv4) or (`inet6` for IPv6).
   - To filter Layer 2 circuit interface traffic, specify the family address type `ccc`.

   The filter and term names can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. Each filter name must be unique. A filter can contain one or more terms, and each term name must be unique within a filter.

2. Configure additional match conditions. For example:

   In this configuration, the filter matches on Layer 2 packets that contain source port 80.

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one from]
user@switch# set source-port 80
```

   In this configuration, the filter matches on VLANs that contain interface ge-0/0/6.0.

```
[edit firewall family inet filter ingress-interface-match-condition term term-one from]
user@switch#set interface ge-0/0/6.0.
```

   You can specify one or more match conditions in a single `from` statement. For a match to occur, the packet must match all the conditions in the term. The `from` statement is optional, but if you include it in a term, it can't be empty. If you omit the `from` statement, all packets are considered to match.

3. If you want to apply a firewall filter to multiple interfaces and be able to see counters specific to each interface, configure the `interface-specific` option:

```
[edit firewall family ethernet-switching filter ingress-port-filter]
user@switch# set interface-specific
```

4. In each firewall filter term, specify the actions to take if the packet matches all the conditions in that term. You can specify an action and action modifiers:

- To specify a filter action, for example, to discard packets that match the conditions of the filter term:

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one then]
user@switch#  set discard
```

You can specify only one action per term (`accept`, `discard`, `flood`, `reject`, `routing-instance`, or `vlan`).

- To specify a filter action, for example, to flood packets that match the MAC address on QFX5100/QFX5110/ QFX5120-32C/QFX5200/QFX5210:

```
[edit firewall family ethernet-switching filter f1 term t2 then]
user@switch# set flood
```

You can configure the ingress port-based firewall filters to flood or discard the following BPDUs by using the destination MAC address as the match condition.

Protocols	Destination Media Access Control (DMAC) Address	Firewall Action
Link Aggregation Control Protocol (LACP)	01:80:c2:00:00:02	Flood/Discard/Count
Link Layer Discovery Protocol (LLDP)	01:80:c2:00:00:0E	Flood/Discard/Count
Extensible Authentication Protocol over LAN (EAPOL)	01:80:c2:00:00:03	Flood/Discard/Count
Spanning Tree Protocol (STP)	01:80:c2:00:00:00	Flood/Discard/Coun
VLAN Spanning Tree Protocol (VSTP)	01:00:0c:cc:cc:cd	Flood/Discard/Count
Cisco Discovery Protocol (CDP)/VLAN Trunk Protocol (VTP)	01:00:0C:cc:cc:cc	Discard/Count
ISIS L1	01:80:c2:00:00:14	Discard/Count
ISIS L2	01:80:c2:00:00:15	Discard/Count

NOTE:

- CDP/VTP, ISIS L1/L2 protocols flood by using the default dynamic filter. Therefore, configuring additional filters for these protocols is not necessary.
- As ingress port-based firewall filters are applied at the port level, only one filter can be applied for a physical interface in the service provider style configuration.
- The native VLAN must be configured to ensure flooding of the untagged BPDUs received on the trunk port. If the native VLAN is not configured, then the untagged BPDUs will be flooded on all the interfaces in the local FPC.
- When IGMP snooping or multicast listener discovery (MLD) snooping is enabled then, the flood functionality does not work.
- When the firewall filter with flood action is applied on an interface and later if the interface goes down, then the BPDUs received on that interface will be flooded if it satisfies the match conditions.

- To specify action modifiers, for example, to count and classify packets to a forwarding class:

```
[edit firewall family ethernet-switching filter ingress-port-filter term term-one then]
user@switch# set count counter-one
user@switch# set forwarding-class expedited-forwarding
user@switch# set loss-priority high
```

You can specify any of the following action modifiers in a `then` statement:

- `analyzer` *`analyzer-name`*—Mirror port traffic to a specified analyzer, which you must configure at the `[ethernet-switching-options]` level.
- `count` *`counter-name`*—Count the number of packets that pass this filter term.

NOTE: We recommend that you configure a counter for each term in a firewall filter, so that you can monitor the number of packets that match the conditions specified in each filter term.

NOTE: On QFX3500 and QFX3600 switches, filters automatically count packets that were dropped in the ingress direction because of cyclic redundancy check (CRC) errors.

- `forwarding-class` *`class`*—Assign packets to a forwarding class.
- `log`—Log the packet header information in the Routing Engine.
- `loss-priority` *`priority`*—Set the priority of dropping a packet.
- `policer` *`policer-name`*—Apply rate-limiting to the traffic.
- `flood`—Flood the packets.
- `syslog`—Log an alert for this packet.

If you omit the `then` statement or don't specify an action, packets matching all the conditions in the `from` statement are accepted. But make sure that you always configure an action in the `then` statement. You can only include one action statement, but can use any combination of action modifiers. For an action or action modifier to take effect, all conditions in the `from` statement must match.

NOTE: The `implicit discard` action applicable to a firewall filter applied to the loopback interface, `lo0`.

Configuring Enhanced Egress Firewall Filters (QFX5110 and QFX5220 Switches)

Due to a hardware limitation, the QFX5110 and QFX5220 switches can only support a maximum of 1000 egress firewall filters (eRACLs). You can increase this number to 2000, by configuring the switch in scaled mode. In this mode, the switch uses ingress TCAM space (IFP) to achieve the higher scale.

To configure the egress filter, specify the family address type (`inet` for IPv4) or (`inet6` for IPv6), filter name, and term name. Include the applicable scaling option for your switch and specify a match condition and action to take if a match occurs. Then apply the filter in the output direction on the interface.

After configuring, modifying, or deleting a scaling option, you must commit the configuration, and the packet forwarding engine (PFE) must be restarted.

To increase the number of egress filters on the QFX5110, include the `egress-to-ingress` option in your configuration. You can add this option under any term. The following is a sample configuration:

```
set firewall family inet filter f1 term t1 from egress-to-ingress
set firewall family inet filter f1 term t1 from source-port 1500
set firewall family inet filter f1 term t1 then accept
set interfaces irb unit 100 family inet filter output f1
```

To increase the number of egress filters on the QFX5220, include the `eracl-scale` option under the `egress-profile` statement. The following is a sample configuration:

NOTE: The `eracl-scale` option comes configured in global mode. When enabled, existing egress filters will be automatically reinstalled in scaled mode.

```
set system packet-forwarding-options firewall eracl-profile eracl-scale
set firewall family inet filter f1 term t1 from source-port 1500
set firewall family inet filter f1 term t1 then accept
set interfaces irb unit 100 family inet filter output f1
```

When you enable scaled mode, these limitations apply:

- You can only apply a filter in the egress direction (traffic exiting the VLAN).
- Only `inet` and `inet6` protocol families are supported.
- Generic Routing Encapsulation (GRE) interfaces are not supported.
- Only use the scaling options for egress firewall filters.
- You cannot apply filters with the same match condition to different egress VLANs or Layer 3 interfaces. The only supported actions are `accept`, `discard`, and `count`.
- Match conditions are programmed in the ingress firewall filter TCAM. This means that any counters attached to the filter counts traffic on any incoming VLANs.

Applying a Firewall Filter to a Port

To apply a firewall filter to a port:

1. Provide a meaningful and descriptive name for the firewall filter. The name is what you use to apply the filter to the port.

   ```
   [edit]
   user@switch# set interfaces ge-0/0/6 description "filter to limit tcp traffic at trunk port
   for employee-vlan"
   ```

2. Apply the filter to the interface, specifying the unit number, family address type (`ethernet-switching`), the direction of the filter (for packets entering the port), and the filter name:

   ```
   [edit]
   user@switch# set ge-0/0/6 unit 0 family ethernet-switching filter input ingress-port-filter
   ```

NOTE: You can apply only one filter to a port in the ingress direction.

Applying a Firewall Filter to a VLAN

NOTE: VLAN firewall filters are not supported on QFX5100, QFX5100 Virtual Chassis, QFX5110, and QFX5120 switches in an EVPN-VXLAN environment.

To apply a firewall filter to a VLAN:

1. Provide a meaningful and descriptive name for the firewall filter. This name is what you use to apply the filter to the VLAN.

```
[edit]
user@switch# set vlans employee-vlan vlan-id 20 description "filter to block rogue devices on
employee-vlan"
```

2. Apply firewall filters to filter packets entering or exiting the VLAN:
   - To apply a filter to match packets entering the VLAN:

```
[edit]
user@switch#  set vlans employee-vlan vlan-id 20 filter input ingress-vlan-rogue-block
```

   - To apply a firewall filter to match packets exiting the VLAN:

```
[edit]
user@switch#  set vlans employee-vlan vlan-id 20 filter output egress-vlan-filter
```

NOTE: You can apply only one filter to a VLAN for a given direction (ingress or egress).

Applying a Firewall Filter to a Layer 3 (Routed) Interface

You can apply a firewall filter to IPv4 and IPv6 interfaces, routed VLAN interfaces (RVI) (also known as an integrated routing and bridging (IRB) interface), and the loopback interface. These are all considered Layer 3 routed interfaces.

NOTE: (QFX5100 and QFX5110 switches) In an EVPN-VXLAN environment, you can use an IRB interface to provide layer 3 connectivity to the switch. To configure an IRB interface, see *Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center*. You can then apply a firewall filter to the IRB interface by following the steps below (only the ingress direction is supported). For a list of supported match conditions, see "Firewall Filter Match Conditions and Actions (QFX5100, QFX5110, QFX5120, QFX5200, EX4600, EX4650)" on page 1732.

NOTE: When you apply a filter to an IRB interface associated with a given VLAN, the filter is executed on any Layer 3 interface with a matching VLAN ID. This is because the filter matches on all Layer 3 interfaces with the corresponding VLAN tag.

To apply a firewall filter to a Layer 3 interface:

1. Provide a meaningful and descriptive name for the firewall filter. This name is what you use to apply the filter to the interface.

```
[edit]
user@switch# set interfaces ge-0/1/6 description "filter to count and monitor traffic on
layer 3 interface"
```

2. Apply the firewall filters.
   - To filter packets entering the interface:

```
[edit]
user@switch# set interfaces ge-0/1/6 unit 0 family inet filter input ingress-router-filter
```

   - To filter packets exiting the interface:

```
[edit]
user@switch# set interfaces ge-0/1/6 unit 0 family inet filter output egress-router-filter
```

     The family address type can either be (`inet` for IPv4) or (`inet6` for IPv6).

NOTE: You can apply only one filter to an interface for a given direction (ingress or egress).

Applying a Firewall Filter to a Layer 2 CCC (QFX10000 Switches)

You can apply firewall filters with count and policer actions on Layer 2 circuit cross-connect (CCC) traffic on QFX10000 switches. This lets you count and monitor the policer activity set at the `[edit firewall family ccc]` hierarchy level.

In this example, `count` is the policer action.

```
set firewall policer traffic-cnt if-exceeding bandwidth-limit 1g
set firewall policer traffic-cnt if-exceeding burst-size-limit 100m
set firewall policer traffic-cnt then loss-priority low
set firewall family ccc filter srTCM-cnt term t1 then policer traffic-cnt
set firewall family ccc filter srTCM-cnt term t1 then count traffic-counter
```

In this example, `discard` is the policer action.

```
set firewall policer discard-traffic if-exceeding bandwidth-limit 1g
set firewall policer discard-traffic if-exceeding burst-size-limit 500m
set firewall policer discard-traffic then discard
set firewall family ccc filter srTCM1 term t1 then policer discard-traffic
```

RELATED DOCUMENTATION

- Overview of Firewall Filters (QFX Series) | **1716**
- Planning the Number of Firewall Filters to Create | **1721**
- Firewall Filter Match Conditions and Actions (QFX and EX Series Switches) | **1731**
- Firewall Filter Match Conditions and Actions (QFX10000 Switches) | **1780**
- Monitoring Firewall Filter Traffic | **827**

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 a logical interface at the `[edit interfaces]` hierarchy level:

```
[edit interfaces]
user@switch# set interface-name unit logical-unit-number family family-name filter (input |
output) filter-name
```

In the `input` statement, specify a firewall filter to be evaluated when packets are received on the interface. Input filters applied to a loopback interface affect only traffic destined for the Routing Engine.

In the `output` statement, specify a filter to be evaluated when packets exit the interface.

NOTE: When you create a loopback interface, it is important to apply an ingress filter to it so the Routing Engine is protected. We recommend that when you apply a filter to the loopback interface `lo0`, you include the `apply-groups` statement. Doing so ensures that the filter is automatically inherited on every loopback interface, including `lo0` and other loopback interfaces.

RELATED DOCUMENTATION

Configuring Firewall Filters | 1826

Overview of MPLS Firewall Filters on Loopback Interface

IN THIS SECTION

- Benefits of Adding MPLS Firewall Filters on the Loopback Interface | 1836
- Guidelines and Limitations | 1836

Although all interfaces are important, the loopback interface might be the most important because it is the link to the Routing Engine, which runs and manages all the routing protocols. The loopback interface is a gateway for all the control traffic that enters the Routing Engine of the switch. You can control this

traffic by configuring a firewall filter on the loopback interface (lo0) on `family mpls`. Loopback firewall filters affect only traffic destined for the Routing Engine CPU. You can apply a loopback firewall filter only in the *ingress* direction (packets entering the interface). Starting with Junos OS Release 19.2R1, you can apply an MPLS firewall filter to a loopback interface on a label switch router (LSR) on QFX5100, QFX5110, QFX5200, and QFX5210 switches.

When you configure an MPLS firewall filter, you define filtering criteria (*terms, with match conditions*) for the packets and an *action* for the switch to take if the packets match the filtering criteria. Because you apply the filter to a loopback interface, you must explicitly specify the time to live (TTL) match condition under `family mpls` and set its TTL value to 1 (`ttl=1`). The TTL is an 8-bit (IPv4) header field that signifies the remaining time an IP packet has left before its life ends and is dropped. You can also match packets with other MPLS qualifiers such as `label`, `exp`, Layer 4 `source port`, and Layer 4 `destination port`.

Benefits of Adding MPLS Firewall Filters on the Loopback Interface

- Protects the Routing Engine by ensuring that it accepts traffic only from trusted networks.
- Helps protect the Routing Engine from denial-of-service attacks.
- Gives you the flexibility to match packets on the source port and destination port. For example, if you run a traceroute, you can selectively filter traffic by choosing either TCP or UDP.

Guidelines and Limitations

- You can apply a loopback firewall filter only in the *ingress* direction
- Only MPLS fields `label`, `exp`, `ttl=1` and Layer 4 fields `tcp` and `udp` port numbers are supported.
- Only `accept`, `discard`, and `count` actions are supported.
- You must explicitly specify `ttl=1` under `family mpls` to match on TLL packets.
- Filters applied on the loopback interface cannot be matched on the destination port (inner payload) of an IPv6 packet.
- You cannot apply a filter on packets that have more than two MPLS labels.
- You cannot specify a port range for TCP or UDP match conditions.
- Only 255 firewall terms are supported.

Release History Table

Release	Description
19.2R1	Starting with Junos OS Release 19.2R1, you can apply an MPLS firewall filter to a loopback interface on a label switch router (LSR) on QFX5100, QFX5110, QFX5200, and QFX5210 switches.

Configuring MPLS Firewall Filters and Policers on Switches

IN THIS SECTION

- Configuring an MPLS Firewall Filter | **1837**
- Applying an MPLS Firewall Filter to an MPLS Interface | **1838**
- Applying an MPLS Firewall Filter to a Loopback Interface | **1838**
- Configuring Policers for LSPs | **1839**

You can configure firewall filters to filter MPLS traffic. To use an MPLS firewall filter, you must first configure the filter and then apply it to an interface you have configured for forwarding MPLS traffic. You can also configure a policer for the MPLS filter to police (that is, rate-limit) the traffic on the interface to which the filter is attached.

When you configure an MPLS firewall filter, you define the filtering criteria (terms, with match conditions) and an action for the switch to take if the packets match the filtering criteria.

NOTE: You can only configure MPLS filters in the ingress direction. Egress MPLS firewall filters are not supported.

Configuring an MPLS Firewall Filter

To configure an MPLS firewall filter:

1. Configure the filter name, term name, and at least one match condition—for example, match on MPLS packets with EXP bits set to either 0 or 4:

```
[edit firewall family mpls]
user@switch# set filter ingress-exp-filter term term-one from exp 0,4
```

2. In each firewall filter term, specify the actions to take if the packet matches all the conditions in that term—for example, count MPLS packets with EXP bits set to either 0 or 4:

```
[edit firewall family mpls filter ingress-exp-filter term term-one then]
user@switch# set count counter0
user@switch# set accept
```

3. When you are finished, follow the steps below to apply the filter to an interface.

Applying an MPLS Firewall Filter to an MPLS Interface

To apply the MPLS firewall filter to an interface you have configured for forwarding MPLS traffic (using the `family mpls` statement at the `[edit interfaces` *interface-name* `unit` *unit-number*`]` hierarchy level):

NOTE: You can apply firewall filters only to filter MPLS packets that enter an interface.

1. Apply the firewall filter to an MPLS interface—for example, apply the firewall filter to interface xe-0/0/5:

```
[edit interfaces]
user@switch# set xe-0/0/5 unit 0 family mpls filter input ingress-exp-filter
```

2. Review your configuration and issue the `commit` command:

```
[edit interfaces]
user@switch# commit
commit complete
```

Applying an MPLS Firewall Filter to a Loopback Interface

To apply an MPLS firewall filter to a loopback interface (lo0):

1. First, specify the packet format by using the *packet-format-match* command. You must restart the PFE every time you configure this command.

2. Configure the firewall filter match conditions and actions as described in "Configuring an MPLS Firewall Filter" on page 1837. You must explicitly set the TTL match condition to (`ttl=1`). You can also match packets with other MPLS qualifiers such as `label`, `exp`, and Layer 4 `source port`, and `destination port`.
3. Apply the filter to the loopback interface as an input filter.

```
[edit interfaces]
user@switch# set lo0 unit 0 family mpls filter input ingress-exp-filter
```

4. Review your configuration and issue the `commit` command:

```
[edit interfaces]
user@switch# commit
commit complete
```

The following is an example configuration.

```
set groups lo_mpls_filter interfaces lo0 unit 0 family mpls filter input mpls_lo
set groups lo_mpls_filter firewall family mpls filter mpls_lo term mpls_lo_term from ttl 1
set groups lo_mpls_filter firewall family mpls filter mpls_lo term mpls_lo_term from ip-version ipv4
protocol udp source-port 10
set groups lo_mpls_filter firewall family mpls filter mpls_lo term mpls_lo_term from ip-version ipv4
protocol udp destination-port 11
set groups lo_mpls_filter firewall family mpls filter mpls_lo term mpls_lo_term then count c1
set groups lo_mpls_filter firewall family mpls filter mpls_lo term mpls_lo_term then accept
```

Configuring Policers for LSPs

Starting with Junos OS 13.2X51-D15, you can send traffic matched by an MPLS filter to a two-color policer or three-color policer. MPLS LSP policing allows you to control the amount of traffic forwarded through a particular LSP. Policing helps to ensure that the amount of traffic forwarded through an LSP never exceeds the requested bandwidth allocation. LSP policing is supported on regular LSPs, LSPs configured with DiffServ-aware traffic engineering, and multiclass LSPs. You can configure multiple policers for each multiclass LSP. For regular LSPs, each LSP policer is applied to all of the traffic traversing the LSP. The policer's bandwidth limitations become effective as soon as the total sum of traffic traversing the LSP exceeds the configured limit.

You configure the multiclass LSP and DiffServ-aware traffic engineering LSP policers in a filter. The filter can be configured to distinguish between the different class types and apply the relevant policer to each class type. The policers distinguish between class types based on the EXP bits.

You configure LSP policers under the `family any` filter. The `family any` filter is used because the policer is applied to traffic entering the LSP. This traffic might be from different families: IPv6, MPLS, and so on. You do not need to know what sort of traffic is entering the LSP, as long as the match conditions apply to all types of traffic.

When configuring MPLS LSP policers, be aware of the following limitations:

- LSP policers are supported for packet LSPs only.
- LSP policers are supported for unicast next hops only. Multicast next hops are not supported.
- The LSP policer runs before any output filters.
- Traffic sourced from the Routing Engine (for example, ping traffic) does not take the same forwarding path as transit traffic. This type of traffic cannot be policed.

Configuring MPLS Firewall Filters and Policers on Routers

IN THIS SECTION

- Configuring MPLS Firewall Filters | **1840**
- Examples: Configuring MPLS Firewall Filters | **1841**
- Configuring Policers for LSPs | **1842**
- Example: Configuring an LSP Policer | **1844**
- Configuring Automatic Policers | **1845**
- Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets | **1849**

You can configure an MPLS firewall filter to count packets based on the EXP bits for the top-level MPLS label in a packet. You can also configure policers for MPLS LSPs.

The following sections discuss MPLS firewall filters and policers:

Configuring MPLS Firewall Filters

You can configure an MPLS firewall filter to count packets based on the EXP bits for the top-level MPLS label in a packet. You can then apply this filter to a specific interface. You can also configure a policer for the MPLS filter to police (that is, rate-limit) the traffic on the interface to which the filter is attached. You cannot apply MPLS firewall filters to Ethernet (fxp0) or loopback (lo0) interfaces.

You can configure the following match criteria attributes for MPLS filters at the `[edit firewall family mpls filter filter-name term term-name from]` hierarchy level:

- `exp`
- `exp-except`

These attributes can accept EXP bits in the range 0 through 7. You can configure the following choices:

- A single EXP bit—for example, `exp 3;`
- Several EXP bits—for example, `exp 0, 4;`
- A range of EXP bits—for example, `exp [0-5];`

If you do not specify a match criterion (that is, you do not configure the `from` statement and use only the `then` statement with the `count` action keyword), all the MPLS packets passing through the interface on which the filter is applied will be counted.

You also can configure any of the following action keywords at the `[edit firewall family mpls filter filter-name term term-name then]` hierarchy level:

- `count`
- `accept`
- `discard`
- `next`
- `policer`

For more information about how to configure firewall filters, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide. For more information about how to configure interfaces, see the Junos OS Network Interfaces Library for Routing Devices and the Junos OS Services Interfaces Library for Routing Devices.

Examples: Configuring MPLS Firewall Filters

The following examples illustrate how you might configure an MPLS firewall filter and then apply the filter to an interface. This filter is configured to count MPLS packets with EXP bits set to either 0 or 4.

The following shows a configuration for an MPLS firewall filter:

```
[edit firewall]
family mpls {
    filter expf {
```

```
        term expt0 {
            from {
                exp 0,4;
            }
            then {
                count counter0;
                accept;
            }
        }
    }
}
```

The following shows how to apply the MPLS firewall filter to an interface:

```
[edit interfaces]
so-0/0/0 {
    mtu 4474;
    encapsulation ppp;
    sonet-options {
        fcs 32;
    }
    unit 0 {
        point-to-point;
        family mpls {
            filter {
                input expf;
                output expf;
            }
        }
    }
}
```

The MPLS firewall filter is applied to the input and output of an interface (see the `input` and `output` statements in the preceding example).

Configuring Policers for LSPs

MPLS LSP policing allows you to control the amount of traffic forwarded through a particular LSP. Policing helps to ensure that the amount of traffic forwarded through an LSP never exceeds the requested bandwidth allocation. LSP policing is supported on regular LSPs, LSPs configured with DiffServ-aware traffic engineering, and multiclass LSPs. You can configure multiple policers for each multiclass LSP. For regular LSPs, each LSP policer is applied to all of the traffic traversing the LSP. The

policer's bandwidth limitations become effective as soon as the total sum of traffic traversing the LSP exceeds the configured limit.

NOTE: The PTX10003 router only supports regular LSPs.

You configure the multiclass LSP and DiffServ-aware traffic engineering LSP policers in a filter. The filter can be configured to distinguish between the different class types and apply the relevant policer to each class type. The policers distinguish between class types based on the EXP bits.

You configure LSP policers under the `family any` filter. The `family any` filter is used because the policer is applied to traffic entering the LSP. This traffic might be from different families: IPv6, MPLS, and so on. You do not need to know what sort of traffic is entering the LSP, as long as the match conditions apply to all types of traffic.

You can configure only those match conditions that apply across all types of traffic. The following are the supported match conditions for LSP policers:

- `forwarding-class`
- `packet-length`
- `interface`
- `interface-set`

To enable a policer on an LSP, first you need to configure a policing filter and then include it in the LSP configuration. For information about how to configure policers, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

To configure a policer for an LSP, specify a filter by including the `filter` option to the `policing` statement:

```
policing {
    filter filter-name;
}
```

You can include the `policing` statement at the following hierarchy levels:

- [edit protocols mpls label-switched-path *lsp-name*]
- [edit protocols mpls static-label-switched-path *lsp-name*]
- [edit logical-systems *logical-system-name* protocols mpls label-switched-path *lsp-name*]
- [edit logical-systems *logical-system-name* protocols mpls static-label-switched-path *lsp-name*]

LSP Policer Limitations

When configuring MPLS LSP policers, be aware of the following limitations:

- LSP policers are supported for packet LSPs only.
- LSP policers are supported for unicast next hops only. Multicast next hops are not supported.
- LSP policers are not supported on aggregated interfaces.
- The LSP policer runs before any output filters.
- Traffic sourced from the Routing Engine (for example, ping traffic) does not take the same forwarding path as transit traffic. This type of traffic cannot be policed.
- LSP policers work on all T Series routers and on M Series routers that have the Internet Processor II application-specific integrated circuit (ASIC).
- LSP policers are not supported for point-to-multipoint LSPs.

NOTE: Starting with Junos OS Release 12.2R2, on T Series routers only, you can configure an LSP policer for a specific LSP to be shared across different protocol family types. To do so, you must configure the *logical-interface-policer* statement at the `[edit firewall policer policer-name]` hierarchy level.

Example: Configuring an LSP Policer

The following example shows how you can configure a policing filter for an LSP:

```
[edit firewall]
policer police-ct1 {
    if-exceeding {
        bandwidth-limit 50m;
        burst-size-limit 1500;
    }
    then {
        discard;
    }
}
policer police-ct0 {
    if-exceeding {
        bandwidth-limit 200m;
```

```
        burst-size-limit 1500;
    }
    then {
        discard;
    }
}
family any {
    filter bar {
        term discard-ct0 {
            then {
                policer police-ct0;
                accept;
            }
        }
    }
    term discard-ct1 {
        then {
            policer police-ct1;
            accept;
        }
    }
}
```

Configuring Automatic Policers

Automatic policing of LSPs allows you to provide strict service guarantees for network traffic. Such guarantees are especially useful in the context of Differentiated Services for traffic engineered LSPs, providing better emulation for ATM wires over an MPLS network. For more information about Differentiated Services for LSPs, see DiffServ-Aware Traffic Engineering Introduction.

Differentiated Services for traffic engineered LSPs allow you to provide differential treatment to MPLS traffic based on the EXP bits. To ensure these traffic guarantees, it is insufficient to simply mark the traffic appropriately. If traffic follows a congested path, the requirements might not be met.

LSPs are guaranteed to be established along paths where enough resources are available to meet the requirements. However, even if the LSPs are established along such paths and are marked properly, these requirements cannot be guaranteed unless you ensure that no more traffic is sent to an LSP than there is bandwidth available.

It is possible to police LSP traffic by manually configuring an appropriate filter and applying it to the LSP in the configuration. However, for large deployments it is cumbersome to configure thousands of different filters. Configuration groups cannot solve this problem either, since different LSPs might have

different bandwidth requirements, requiring different filters. To police traffic for numerous LSPs, it is best to configure automatic policers.

When you configure automatic policers for LSPs, a policer is applied to all of the LSPs configured on the router. However, you can disable automatic policing on specific LSPs.

NOTE: When you configure automatic policers for DiffServ-aware traffic engineering LSP, GRES is not supported.

NOTE: You cannot configure automatic policing for LSPs carrying CCC traffic.

The following sections describe how to configure automatic policers for LSPs:

Configuring Automatic Policers for LSPs

To configure automatic policers for standard LSPs (neither DiffServ-aware traffic engineered LSPs nor multiclass LSPs), include the `auto-policing` statement with either the `class all` *`policer-action`* option or the `class ct0` *`policer-action`* option:

```
auto-policing {
    class all policer-action;
    class ct0 policer-action;
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols mpls]`
- `[edit logical-systems` *`logical-system-name`* `protocols mpls]`

You can configure the following policer actions for automatic policers:

- `drop`—Drop all packets.
- `loss-priority-high`—Set the packet loss priority (PLP) to high.
- `loss-priority-low`—Set the PLP to low.

These policer actions are applicable to all types of LSPs. The default policer action is to do nothing.

Automatic policers for LSPs police traffic based on the amount of bandwidth configured for the LSPs. You configure the bandwidth for an LSP using the `bandwidth` statement at the `[edit protocols mpls label-`

switched-path *lsp-path-name*] hierarchy level. If you have enabled automatic policers on a router, change the bandwidth configured for an LSP, and commit the revised configuration, the change does not take affect on the active LSPs. To force the LSPs to use the new bandwidth allocation, issue a `clear mpls lsp` command.

NOTE: You cannot configure automatic policers for LSPs that traverse aggregated interfaces or Multilink Point-to-Point Protocol (MLPPP) interfaces.

Configuring Automatic Policers for DiffServ-Aware Traffic Engineering LSPs

To configure automatic policers for DiffServ-aware traffic engineering LSPs and for multiclass LSPs, include the `auto-policing` statement:

```
auto-policing {
    class all policer-action;
    class ctnumber policer-action;
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols mpls]`
- `[edit logical-systems logical-system-name protocols mpls]`

You include either the `class all` *policer-action* statement or a `class ct`*number policer-action* statement for each of one or more classes (you can configure a different policer action for each class). For a list of the actions that you can substitute for the *policer-action* variable, see "Configuring Automatic Policers for LSPs" on page 1846. The default policer action is to do nothing.

NOTE: You cannot configure automatic policers for LSPs that traverse aggregated interfaces or MLPPP interfaces.

Configuring Automatic Policers for Point-to-Multipoint LSPs

You can configure automatic policers for point-to-multipoint LSPs by including the `auto-policing` statement with either the `class all` *policer-action* option or the `class ct0` *policer-action* option. You only need to configure the `auto-policing` statement on the primary point-to-multipoint LSP (for more information on primary point-to-multipoint LSPs, see Configuring the Primary Point-to-Multipoint LSP). No additional configuration is required on the subLSPs for the point-to-multipoint LSP. Point-to-

multipoint automatic policing is applied to all branches of the point-to-multipoint LSP. In addition, automatic policing is applied to any local VRF interfaces that have the same forwarding entry as a point-to-multipoint branch. Feature parity for automatic policers for MPLS point-to-multipoint LSPs on the Junos Trio chipset is supported in Junos OS Releases 11.1R2, 11.2R2, and 11.4.

The automatic policer configuration for point-to-multipoint LSPs is identical to the automatic policer configuration for standard LSPs. For more information, see "Configuring Automatic Policers for LSPs" on page 1846.

Disabling Automatic Policing on an LSP

When you enable automatic policing, all of the LSPs on the router or logical system are affected. To disable automatic policing on a specific LSP on a router where you have enabled automatic policing, include the `policing` statement with the `no-auto-policing` option:

```
policing no-auto-policing;
```

You can include this statement at the following hierarchy levels:

- [edit protocols mpls label-switched-path *lsp-name*]
- [edit logical-systems *logical-system-name* protocols mpls label-switched-path *lsp-name*]

Example: Configuring Automatic Policing for an LSP

Configure automatic policing for a multiclass LSP, specifying different actions for class types `ct0`, `ct1`, `ct2`, and `ct3`.

```
[edit protocols mpls]
diffserv-te {
    bandwidth-model extended-mam;
}
auto-policing {
    class ct1 loss-priority-low;
    class ct0 loss-priority-high;
    class ct2 drop;
    class ct3 loss-priority-low;
}
traffic-engineering bgp-igp;
label-switched-path sample-lsp {
    to 3.3.3.3;
    bandwidth {
```

```
        ct0 11;
        ct1 1;
        ct2 1;
        ct3 1;
    }
}
interface fxp0.0 {
    disable;
}
interface t1-0/5/3.0;
interface t1-0/5/4.0;
```

Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets

You can selectively set the DiffServ code point (DSCP) field of MPLS-tagged IPv4 and IPv6 packets to 0 without affecting output queue assignment, and continue to set the MPLS EXP field according to the configured rewrite table, which is based on forwarding classes. You can accomplish this by configuring a firewall filter for the MPLS-tagged packets.

Configuring MPLS Firewall Filters and Policers

IN THIS SECTION

- Configuring MPLS Firewall Filters | **1849**
- Examples: Configuring MPLS Firewall Filters | **1850**
- Configuring Policers for LSPs | **1851**

You can configure an MPLS firewall filter to count packets based on the EXP bits for the top-level MPLS label in a packet. You can also configure policers for MPLS LSPs.

The following sections discuss MPLS firewall filters and policers:

Configuring MPLS Firewall Filters

You can configure an MPLS firewall filter to count packets based on the EXP bits for the top-level MPLS label in a packet. You can then apply this filter to a specific interface on input or output. You can also

configure a policer for the MPLS filter to police (that is, rate-limit) the traffic on the interface to which the filter is attached. You cannot apply MPLS firewall filters to loopback interfaces.

You can configure the following match conditions for MPLS filters at the `[edit firewall family mpls filter filter-name term term-name from]` hierarchy level:

- `exp`
- `label`

These `exp` match condition can accept EXP bits in the range 0 through 7. You can configure the following choices:

- A single EXP bit—for example, `exp 3;`
- Several EXP bits—for example, `exp 0, 4;`
- A range of EXP bits—for example, `exp [0-5];`

The `label` match condition can accept a range of values from 0 to 1048575.

If you do not specify a match criterion (that is, you do not configure the `from` statement and use only the `then` statement with the `count` action keyword), all the MPLS packets passing through the interface on which the filter is applied will be counted.

You also can configure any of the following action keywords at the `[edit firewall family mpls filter filter-name term term-name then]` hierarchy level:

- `accept`
- `count`
- `discard`
- `policer`
- `three-color-policer`

Examples: Configuring MPLS Firewall Filters

The following examples illustrate how you might configure an MPLS firewall filter and then apply the filter to an interface. This filter is configured to count MPLS packets with EXP bits set to either 0 or 4.

The following shows a configuration for an MPLS firewall filter:

```
[edit firewall]
family mpls {
```

```
    filter expf {
        term expt0 {
            from {
                exp 0,4;
            }
            then {
                count counter0;
                accept;
            }
        }
    }
}
```

Configuring Policers for LSPs

MPLS LSP policing allows you to control the amount of traffic forwarded through a particular LSP. Policing helps to ensure that the amount of traffic forwarded through an LSP never exceeds the requested bandwidth allocation. LSP policing is supported on regular LSPs, LSPs configured with DiffServ-aware traffic engineering, and multiclass LSPs. You can configure multiple policers for each multiclass LSP. For regular LSPs, each LSP policer is applied to all of the traffic traversing the LSP. The policer's bandwidth limitations become effective as soon as the total sum of traffic traversing the LSP exceeds the configured limit.

You configure the multiclass LSP and DiffServ-aware traffic engineering LSP policers in a filter. The filter can be configured to distinguish between the different class types and apply the relevant policer to each class type. The policers distinguish between class types based on the EXP bits.

You configure LSP policers under the `family any` filter. The `family any` filter is used because the policer is applied to traffic entering the LSP. This traffic might be from different families: IPv6, MPLS, and so on. You do not need to know what sort of traffic is entering the LSP, as long as the match conditions apply to all types of traffic.

LSP Policer Limitations

When configuring MPLS LSP policers, be aware of the following limitations:

- LSP policers are supported for packet LSPs only.
- LSP policers are supported for unicast next hops only. Multicast next hops are not supported.
- LSP policers are not supported on aggregated interfaces.
- The LSP policer runs before any output filters.

- Traffic sourced from the Routing Engine (for example, ping traffic) does not take the same forwarding path as transit traffic. This type of traffic cannot be policed.
-

RELATED DOCUMENTATION

Overview of Policers | 2198

Understanding How a Firewall Filter Tests a Protocol

When examining match conditions in a *firewall filter*, a switch tests only the fields that you specify. It does not implicitly test any fields that you do not explicitly configure. For example, if you specify a match condition of `source-port ssh`, there is no implied test to determine if the protocol is TCP. In this case, the switch considers any packet that has a value of `22` (decimal) in the 2-byte field that follows a *presumed* IP header to be a match. To ensure that the term matches on TCP packets, you also specify an `ip-protocol tcp` match condition.

For the following match conditions, you should explicitly specify the protocol match condition in the same term:

- `destination-port`—Specify protocol `tcp` or protocol `udp`.
- `icmp-code`—Specify protocol `icmp` and `icmp-type`.
- `icmp-type`—Specify protocol `icmp` or protocol `icmp6`.
- `source-port`—Specify protocol `tcp` or protocol `udp`.
- `tcp-flags`—Specify protocol `tcp`.

RELATED DOCUMENTATION

Understanding Firewall Filter Match Conditions | 856

Configuring Firewall Filters | 1826

Understanding Firewall Filter Processing Points for Bridged and Routed Packets

You apply firewall filters at multiple processing points in the forwarding path. At each processing point, the action to be taken on a packet is determined by the configuration of the filter and the results of the lookup in the forwarding or routing table.

For both bridged (Layer 2) unicast packets and routed (Layer 3) unicast packets, firewall filters are applied in the prescribed order shown below (assuming that each filter is present and a packet is accepted by each one).

Bridged packets:

1. Ingress port filter
2. Ingress VLAN filter
3. Egress VLAN filter
4. Egress port filter

Routed packets:

1. Ingress port firewall filter
2. Ingress VLAN firewall filter (Layer 2 CoS)
3. Ingress router firewall filter (Layer 3 CoS)
4. Egress router firewall filter
5. Egress VLAN firewall filter
6. Egress port filter

NOTE: MAC learning occurs before filters are applied, so switches learn the MAC addresses of packets that are dropped by ingress filters.

RELATED DOCUMENTATION

Overview of Firewall Filters (QFX Series) | **1716**

Understanding How Firewall Filters Control Packet Flows | **779**

Configuring Firewall Filters | **1826**

Understanding Filter-Based Forwarding

IN THIS SECTION

- Benefits of Filter-Based Forwarding | 1854

For IPv4 or IPv6 traffic, you can use firewall filters in conjunction with virtual routing instances to specify different routes for packets to travel in their networks. This feature is called filter-based forwarding (FBF), and is also known as policy-based routing (PBR).

You might want to use FBF to route specific types of traffic through a firewall or other security device before the traffic continues on its path. You can also use FBF to give certain types of traffic preferential treatment. For example, you might want to ensure that the highest-priority traffic is forwarded over a 40-Gigabit Ethernet link.

To set up FBF, you specify a firewall filter match condition and action and then specify the virtual routing instance to send packets to.

NOTE: You can create as many as 128 filters or terms that direct packets to a given virtual routing instance.
(QFX5100, QFX5110, QFX5200 switches) Starting in Junos OS Release 19.1R1, filter-based forwarding is supported on IPv6 interfaces (ingress direction only).

Benefits of Filter-Based Forwarding

- Allows you to have more control over load balancing than dynamic routing protocols typically provide.

Release History Table

Release	Description
19.1R1	(QFX5100, QFX5110, QFX5200 switches) Starting in Junos OS Release 19.1R1, filter-based forwarding is supported on IPv6 interfaces (ingress direction only).

RELATED DOCUMENTATION

Overview of Firewall Filters (QFX Series) | 1716

Firewall Filter Match Conditions and Actions (EX4400, EX4600, EX4650, QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, QFX5700) | 1732

Understanding Virtual Router Routing Instances

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device

IN THIS SECTION

- Requirements | 1855
- Overview and Topology | 1855
- Configuration | 1856
- Verification | 1859

This example describes how to set up filter-based forwarding on EX Series switches or a QFX10000. You can configure filter-based forwarding by using a firewall filter to forward matched traffic to a specific virtual routing instance.

Requirements

This example applies to both EX Series switches running Junos OS Release 9.4 or later, and QFX10000 switches running Junos OS Release 15.1X53-D10 or later.

Overview and Topology

In this example, we create a firewall filter to match traffic being sent from one application server to another according to the destination address (192.168.0.1) of packets egressing the source application server. Matching packets are routed to a virtual routing instance which forwards the traffic to a security device, which then forwards the traffic on to the destination application server.

NOTE: Filter-based forwarding does not work with IPv6 interfaces on some Juniper switches.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1856
- Procedure | 1857
- Results | 1858

To configure filter-based forwarding:

CLI Quick Configuration

To use this example on your own device, copy the following commands into a text file, remove the line breaks, and change the necessary details to fit your configuration. Then copy and paste the commands into your CLI at the `[edit]` hierarchy level.

```
[edit]
                    set interfaces xe-0/0/0 unit 0 family inet address
10.1.0.1/24
                    set interfaces xe-0/0/3 unit 0 family inet address
10.1.3.1/24
                    set firewall family inet filter f1 term t1 from source-address
10.1.0.50/32
                    set firewall family inet filter f1 term t1 from protocol
tcp
                    set interfaces xe-0/0/0 unit 0 family inet filter input f1
                    set routing-instances vrf01 instance-type virtual-router
                    set routing-instances vrf01 interface xe-0/0/3.0
                    set routing-instances vrf01 routing-options static route 192.168.0.1/24
next-hop 10.1.3.254
                    set firewall family inet filter f1 term t1 then routing-instance
vrf01
```

Procedure

Step-by-Step Procedure

To configure filter-based forwarding:

1. Configure an interface to connect to the application server:

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 0 family inet address 10.1.0.1/24
```

2. Configure an interface to connect to the security device:

```
[edit interfaces]
user@switch# set xe-0/0/3 unit 0 family inet address 10.1.3.1/24
```

3. Create a firewall filter that matches packets based on the address of the application server that the traffic will be sent from. Also configure the filter so that it matches only TCP packets:

```
[edit firewall]
user@switch# set family inet filter f1 term t1 from source-address 10.1.0.50/32
user@switch# set firewall family inet filter f1 term t1 from protocol
tcp
```

4. Apply the filter to the interface that connects to the source application server and configure it to match incoming packets:

```
[edit interfaces]
user@switch# set xe-0/0/0 unit 0 family inet filter input f1
```

5. Create a virtual router:

```
[edit]
user@switch# set routing-instances vrf01 instance-type virtual-router
```

6. Associate the virtual router with the interface that connects to the security device:

```
[edit routing-instances]
user@switch# set vrf01 interface xe-0/0/3.0
```

7. Configure the routing information for the virtual routing instance:

```
[edit routing-instances]
user@switch# set vrf01 routing-options static route 192.168.0.1/24 next-hop 10.1.3.254
```

8. Set the filter to forward packets to the virtual router:

```
[edit firewall]
user@switch# set family inet filter f1 term t1 then routing-instance
vrf01
```

Results

Check the results of the configuration:

```
user@switch> show configuration
interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                filter {
                    input f1;
                }
                address 10.1.0.1/24;
            }
        }
    }
    xe-0/0/3 {
        unit 0 {
            family inet {
                address 10.1.3.1/24;
            }
        }
    }
```

```
}
firewall {
    family inet {
        filter f1 {
            term t1 {
                from {
                    source-address {
                        10.1.0.50/32;
                    }
                    protocol tcp;
                }
                then {
                    routing-instance vrf01;
                }
            }
        }
    }
}
    routing-instances {
        vrf01 {
            instance-type virtual-router;
            interface xe-0/0/3.0;
            routing-options {
                static {
                    route 192.168.0.1/24 next-hop 10.1.3.254;
                }
            }
        }
    }
```

Verification

IN THIS SECTION

- Verifying That Filter-Based Forwarding Was Configured | **1860**

To confirm that the configuration is working properly, perform these tasks:

Verifying That Filter-Based Forwarding Was Configured

Purpose

Verify that filter-based forwarding was properly enabled on the switch.

Action

1. Use the `show interfaces filters` command:

```
user@switch>                                    show interfaces filters
xe-0/0/0.0
Interface        Admin Link Proto Input Filter         Output Filter
xe-0/0/0.0       up    down inet  fil
```

2. Use the `show route forwarding-table` command:

```
user@switch> show route forwarding-table

Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            user     1 0:12:f2:21:cf:0    ucst   331     4 me0.0
default            perm     0                    rjct    36     3
0.0.0.0/32         perm     0                    dscd    34     1
10.1.0.0/24        ifdn     0                    rslv   613     1 xe-0/0/0.0
10.1.0.0/32        iddn     0 10.1.0.0           recv   611     1 xe-0/0/0.0
10.1.0.1/32        user     0                    rjct    36     3
10.1.0.1/32        intf     0 10.1.0.1           locl   612     2
10.1.0.1/32        iddn     0 10.1.0.1           locl   612     2
10.1.0.255/32      iddn     0 10.1.0.255         bcst   610     1 xe-0/0/0.0
10.1.1.0/26        ifdn     0                    rslv   583     1 vlan.0
10.1.1.0/32        iddn     0 10.1.1.0           recv   581     1 vlan.0
10.1.1.1/32        user     0                    rjct    36     3
10.1.1.1/32        intf     0 10.1.1.1           locl   582     2
10.1.1.1/32        iddn     0 10.1.1.1           locl   582     2
10.1.1.63/32       iddn     0 10.1.1.63          bcst   580     1 vlan.0
255.255.255.255/32 perm     0                    bcst    32     1

Routing table: vrf01.inet
Internet:
```

```
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   559     2
0.0.0.0/32         perm     0                    dscd   545     1
10.1.3.0/24        ifdn     0                    rslv   617     1 xe-0/0/3.0
10.1.3.0/32        iddn     0 10.1.3.0           recv   615     1 xe-0/0/3.0
10.1.3.1/32        user     0                    rjct   559     2
192.168.0.1/24     user     0 10.1.3.254         ucst   616     2 xe-0/0/3.0
192.168.0.1/24     user     0 10.1.3.254         ucst   616     2 xe-0/0/3.0
10.1.3.255/32      iddn     0 10.1.3.255         bcst   614     1 xe-0/0/3.0
224.0.0.0/4        perm     0                    mdsc   546     1
224.0.0.1/32       perm     0 224.0.0.1          mcst   529     1
255.255.255.255/32 perm     0                    bcst   543     1

Routing table: default.iso
ISO:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct    60     1

Routing table: vrf01.iso
ISO:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   600     1
```

Meaning

The output indicates that the filter was created on the interface and that the virtual routing instance is forwarding matching traffic to the correct IP address.

RELATED DOCUMENTATION

- Configuring Firewall Filters | **1826**
- Understanding Filter-Based Forwarding | **1854**
- Understanding Virtual Router Routing Instances

Configuring a Firewall Filter to De-Encapsulate GRE or IPIP Traffic

IN THIS SECTION

- Configuring a Filter to De-Encapsulate GRE Traffic | 1862
- Configuring a Filter to De-Encapsulate IPIP Traffic | 1864
- Applying the Filter to an Interface | 1865

Generic routing encapsulation (GRE) and IP over IP (IPIP) both provide a private, secure path for transporting packets through a network by encapsulating (or tunneling) the packets. The tunneling is performed by tunnel endpoints that encapsulate or de-encapsulate traffic.

You can use a firewall filter to de-encapsulate tunnel traffic on the switch. This feature provides significant benefits in terms of scalability, performance, and flexibility because you don't need to create a tunnel interface to perform the de-encapsulation. For example, you can terminate many tunnels from multiple source IP addresses with one firewall term.

NOTE: The EX4600, QFX5100 and OCX switches support as many as 512 GRE tunnels, including tunnels created with a firewall filter. That is, you can create a total of 512 GRE tunnels, regardless of which method you use.

Configuring a Filter to De-Encapsulate GRE Traffic

To configure a firewall filter to de-encapsulate GRE traffic:

1. Create an IPv4 firewall filter and (optionally) specify a source address for the tunnel:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from source-address
address
```

You must create an IPv4 filter by using `family inet` because the outer header of a GRE packet must be IPv4. If you specify a source address, it should be an address on a device that will encapsulate traffic into GRE packets.

NOTE: To terminate many tunnels from multiple source IP addresses with one firewall term, do not configure a source address. In this case, the filter will de-encapsulate any GRE packets received by the interface that you apply the filter to.

2. Specify a destination address for the tunnel:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from destination-
address address
```

This should be an address on an interface of the switch on which you want the tunnel or tunnels to terminate and the GRE packets to be de-encapsulated. You should also configure this address as a tunnel endpoint on all the tunnel source routers that you want to form tunnels with on the switch.

3. Specify that the filter should match and accept GRE traffic:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from protocol gre
```

4. Specify that the filter should de-encapsulate GRE traffic:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name then decapsulate gre
```

Based on the configuration you have performed so far, the switch forwards the de-encapsulated packets by comparing the inner header to the default routing table (`inet0`). If you want the switch to use a virtual routing instance to forward the de-encapsulated packets, perform the following steps:

5. Specify the name of the virtual routing instance:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name then decapsulate
routing-instance instance-name
```

6. Specify that the virtual routing instance is a virtual router:

```
[edit ]
user@switch# set routing-instances instance-name instance-type virtual-router
```

7. Specify the interfaces that belong to the virtual router:

```
[edit ]
user@switch# set routing-instances instance-name interface interface-name
```

Configuring a Filter to De-Encapsulate IPIP Traffic

To configure a firewall filter to de-encapsulate IPIP traffic::

1. Create an IPv4 firewall filter and (optionally) specify a source address for the tunnel:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from source-address
address
```

You must create an IPv4 filter by using `family inet` because the outer header of an IPIP packet must be IPv4. If you specify a source address, it should be an address on a device that will encapsulate traffic into IPIP packets.

NOTE: To terminate many tunnels from multiple source IP addresses with one firewall term, do not configure a source address. In this case, the filter will de-encapsulate any IPIP packets received by the interface that you apply the filter to.

2. Specify a destination address for the tunnel:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from destination-
address address
```

This should be an address on an interface of the switch on which you want the tunnel or tunnels to terminate and the IPIP packets to be de-encapsulated. You should also configure this address as a tunnel endpoint on all the tunnel source routers that you want to form tunnels with on the switch.

3. Specify that the filter should match and accept IPIP traffic:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name from protocol ipip
```

4. Specify that the filter should de-encapsulate IPIP traffic:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name then decapsulate ipip
```

Based on the configuration you have performed so far, the switch forwards the de-encapsulated packets by comparing the inner header to the default routing table (`inet0`). If you want the switch to use a virtual routing instance to forward the de-encapsulated packets, perform the following steps:

5. Specify the name of the virtual routing instance:

```
[edit ]
user@switch# set firewall family inet filter filter-name term term-name then decapsulate
routing-instance instance-name
```

6. Specify that the virtual routing instance is a virtual router:

```
[edit ]
user@switch# set routing-instances instance-name instance-type virtual-router
```

7. Specify the interfaces that belong to the virtual router:

```
[edit ]
user@switch# set routing-instances instance-name interface interface-name
```

Applying the Filter to an Interface

After you create the firewall filter, you must also apply it to an interface that will receive GRE or IPIP traffic. Be sure to apply it in the input direction. For example, enter

```
[edit ]
user@switch# set interfaces interface-name unit logical-unit-number family inet filter input filter-name
```

Because the outer header of a GRE or IPIP packet must be IPv4, you must apply the filter to an IPv4 interface and specify `family inet`.

RELATED DOCUMENTATION

Understanding Generic Routing Encapsulation

Verifying That Firewall Filters Are Operational

IN THIS SECTION

- Purpose | **1866**
- Action | **1866**
- Meaning | **1867**

Purpose

After you configure and apply firewall filters to ports, VLANs, or Layer 3 interfaces, you can perform the following task to verify that the firewall filters configured on EX Series switches are working properly.

Action

Use the operational mode command to verify that the firewall filters on the switch are working properly:

```
user@switch>                        show firewall
Filter: egress-vlan-watch-employee
Counters:
Name                                                Bytes              Packets
counter-employee-web                                    0                    0
Filter: ingress-port-voip-class-limit-tcp-icmp
Counters:
Name                                                Bytes              Packets
icmp-counter                                            0                    0
Policers:
Name                                               Packets
icmp-connection-policer                                 0
tcp-connection-policer                                  0
Filter: ingress-vlan-rogue-block
Filter: ingress-vlan-limit-guest
```

Meaning

The `show firewall` command displays the names of all firewall filters, policers, and counters that are configured on the switch. For each counter that is specified in a filter configuration, the output field shows the byte count and packet count for the term in which the counter is specified. For each policer that is specified in a filter configuration, the output field shows the packet count for packets that exceed the specified rate limits.

RELATED DOCUMENTATION

- Configuring Firewall Filters (CLI Procedure) | **1638**
- Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**
- Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**
- Monitoring Firewall Filter Traffic | **1867**

Monitoring Firewall Filter Traffic

IN THIS SECTION

- Monitoring Traffic for All Firewall Filters and Policers That Are Configured on the Switch | **1867**
- Monitoring Traffic for a Specific Firewall Filter | **1869**
- Monitoring Traffic for a Specific Policer | **1869**

You can monitor firewall filter traffic on EX Series switches.

Monitoring Traffic for All Firewall Filters and Policers That Are Configured on the Switch

IN THIS SECTION

- Purpose | **1868**
- Action | **1868**

- Meaning | **1868**

Purpose

Perform the following task to monitor the number of packets and bytes that matched the firewall filters and monitor the number of packets that exceeded policer rate limits:

Action

Use the operational mode command:

```
user@switch>                          show firewall
Filter: egress-vlan-watch-employee
Counters:
Name                                                Bytes              Packets
counter-employee-web                                 3348                   27
Filter: ingress-port-voip-class-limit-tcp-icmp
Counters:
Name                                                Bytes              Packets
icmp-counter                                         4100                   49
Policers:
Name                                               Packets
icmp-connection-policer                                  0
tcp-connection-policer                                   0
Filter: ingress-vlan-rogue-block
Filter: ingress-vlan-limit-guest
```

Meaning

The `show firewall` command displays the names of all firewall filters, policers, and counters that are configured on the switch. The output fields show byte and packet counts for counters and packet count for policers.

Monitoring Traffic for a Specific Firewall Filter

IN THIS SECTION

- Purpose | 1869
- Action | 1869
- Meaning | 1869

Purpose

Perform the following task to monitor the number of packets and bytes that matched a firewall filter and monitor the number of packets that exceeded the policer rate limits.

Action

Use the operational mode command:

```
user@switch> show firewall filter ingress-vlan-rogue-block
Filter: ingress-vlan-rogue-block
Counters:
Name                                                Bytes              Packets
rogue-counter                                        2308                   20
```

Meaning

The `show firewall filter` *`filter-name`* command displays the name of the firewall filter, the packet and byte count for all counters configured with the filter, and the packet count for all policers configured with the filter.

Monitoring Traffic for a Specific Policer

IN THIS SECTION

- Purpose | 1870
- Action | 1870

- Meaning | 1870

Purpose

Perform the following task to monitor the number of packets that exceeded policer rate limits:

Action

Use the operational mode command:

```
user@switch>                          show policer tcp-connection-policer
Filter: ingress-port-voip-class-limit-tcp-icmp
Policers:
Name                                                Packets
tcp-connection-policer                                    0
```

Meaning

The `show policer` *`policer-name`* command displays the name of the firewall filter that specifies the policer-action and displays the number of packets that exceeded rate limits for the specified filter.

RELATED DOCUMENTATION

Configuring Firewall Filters (CLI Procedure) | 1638

Configuring Policers to Control Traffic Rates (CLI Procedure) | 2215

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Verifying That Firewall Filters Are Operational | 1866

Troubleshooting Firewall Filter Configuration

IN THIS SECTION

- Firewall Filter Configuration Returns a No Space Available in TCAM Message | **1871**
- Filter Counts Previously Dropped Packet | **1874**
- Matching Packets Not Counted | **1875**
- Counter Reset When Editing Filter | **1876**
- Cannot Include loss-priority and policer Actions in Same Term | **1876**
- Cannot Egress Filter Certain Traffic Originating on QFX Switch | **1877**
- Firewall Filter Match Condition Not Working with Q-in-Q Tunneling | **1878**
- Egress Firewall Filters with Private VLANs | **1878**
- Egress Filtering of L2PT Traffic Not Supported | **1879**
- Cannot Drop BGP Packets in Certain Circumstances | **1880**
- Invalid Statistics for Policer | **1881**
- Policers can Limit Egress Filters | **1881**

Use the following information to troubleshoot your firewall filter configuration.

Firewall Filter Configuration Returns a No Space Available in TCAM Message

IN THIS SECTION

- Problem | **1872**
- Solution | **1872**

Problem

Description

When a firewall filter configuration exceeds the amount of available Ternary Content Addressable Memory (TCAM) space, the system returns the following `syslogd` message:

```
No space available in tcam.
Rules for filter filter-name will not be installed.
```

A switch returns this message during the commit operation if the firewall filter that has been applied to a port, VLAN, or Layer 3 interface exceeds the amount of space available in the TCAM table. The filter is not applied, but the commit operation for the firewall filter configuration is completed in the CLI module.

Solution

When a firewall filter configuration exceeds the amount of available TCAM table space, you must configure a new firewall filter with fewer filter terms so that the space requirements for the filter do not exceed the available space in the TCAM table.

You can perform either of the following procedures to correct the problem:

To delete the filter and its binding and apply the new smaller firewall filter to the same binding:

1. Delete the filter and its binding to ports, VLANs, or Layer 3 interfaces. For example:

```
[edit]
user@switch# delete firewall family ethernet-switching filter ingress-vlan-rogue-block
user@switch# delete vlans employee-vlan description "filter to block rogue devices on
employee-vlan"
user@switch# delete vlans employee-vlan filter input ingress-vlan-rogue-block
```

2. Commit the changes:

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

3. Configure a smaller filter with fewer terms that does not exceed the amount of available TCAM space. For example:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block ...
```

4. Apply (bind) the new firewall filter to a port, VLAN , or Layer 3 interface. For example:

```
[edit]
user@switch# set vlans employee-vlan description "filter to block rogue devices on employee-
vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

5. Commit the changes:

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

To apply a new firewall filter and overwrite the existing binding but not delete the original filter:

1. Configure a firewall filter with fewer terms than the original filter:

```
[edit]
user@switch# set firewall family ethernet-switching filter new-ingress-vlan-rogue-block...
```

2. Apply the firewall filter to the port, VLAN, or Layer 3 interfaces to overwrite the binding of the original filter—for example:

```
[edit]
user@switch# set vlans employee-vlan description "smaller filter to block rogue devices on
employee-vlan"
user@switch# set vlans employee-vlan filter input new-ingress-vlan-rogue-block
```

   Because you can apply no more than one firewall filter per VLAN per direction, the binding of the original firewall filter to the VLAN is overwritten with the new firewall filter `new-ingress-vlan-rogue-block`.

3. Commit the changes:

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

NOTE: The original filter is not deleted and is still available in the configuration.

Filter Counts Previously Dropped Packet

IN THIS SECTION

- Problem | 1874
- Solution | 1875

Problem

Description

If you configure two or more filters in the same direction for a physical interface and one of the filters includes a counter, the counter will be incorrect if the following circumstances apply:

- You configure the filter that is applied to packets first to discard certain packets. For example, imagine that you have a VLAN filter that accepts packets sent to 10.10.1.0/24 addresses and implicitly discards packets sent to any other addresses. You apply the filter to the `admin` VLAN in the output direction, and interface xe-0/0/1 is a member of that VLAN.
- You configure a subsequent filter to accept and count packets that are dropped by the first filter. In this example, you have a port filter that accepts and counts packets sent to 192.168.1.0/24 addresses that is also applied to xe-0/0/1 in the output direction.

The egress VLAN filter is applied first and correctly discards packets sent to 192.168.1.0/24 addresses. The egress port filter is applied next and counts the discarded packets as matched packets. The packets are not forwarded, but the counter displayed by the egress port filter is incorrect.

Remember that the order in which filters are applied depends on the direction in which they are applied, as indicated here:

Ingress filters:

1. Port (Layer 2) filter
2. VLAN filter
3. Router (Layer 3) filter

Egress filters:

1. Router (Layer 3) filter
2. VLAN filter
3. Port (Layer 2) filter

Solution

This is expected behavior.

Matching Packets Not Counted

IN THIS SECTION

- Problem | **1875**
- Solution | **1876**

Problem

Description

If you configure two egress filters with counters for a physical interface and a packet matches both of the filters, only one of the counters includes that packet.

For example:

- You configure an egress port filter with a counter for interface xe-0/0/1.
- You configure an egress VLAN filter with a counter for the `admin`VLAN, and interface xe-0/0/1 is a member of that VLAN.
- A packet matches both filters.

In this case, the packet is counted by only one of the counters even though it matched both filters.

Solution

This is expected behavior.

Counter Reset When Editing Filter

IN THIS SECTION

- Problem | 1876
- Solution | 1876

Problem

Description

If you edit a firewall filter term, the value of any counter associated with any term in the same filter is set to 0, including the implicit counter for any policer referenced by the filter. Consider the following examples:

- Assume that your filter has `term1`, `term2`, and `term3`, and each term has a counter that has already counted matching packets. If you edit any of the terms in any way, the counters for all the terms are reset to 0.
- Assume that your filter has `term1` and `term2`. Also assume that `term2` has a `policer` action modifier and the implicit counter of the policer has already counted 1000 matching packets. If you edit `term1` or `term2` in any way, the counter for the policer referenced by `term2` is reset to 0.

Solution

This is expected behavior.

Cannot Include loss-priority and policer Actions in Same Term

IN THIS SECTION

- Problem | 1877
- Solution | 1877

Problem

Description

You cannot include both of the following actions in the same firewall filter term in a QFX Series switch:

- `loss-priority`
- `policer`

If you do so, you see the following error message when you attempt to commit the configuration: “cannot support policer action if loss-priority is configured.”

Solution

This is expected behavior.

Cannot Egress Filter Certain Traffic Originating on QFX Switch

IN THIS SECTION

- Problem | **1877**
- Solution | **1877**

Problem

Description

On a QFX Series switch, you cannot filter certain traffic with a firewall filter applied in the output direction if the traffic originates on the QFX switch. This limitation applies to control traffic for protocols such as ICMP (ping), STP, LACP, and so on.

Solution

This is expected behavior.

Firewall Filter Match Condition Not Working with Q-in-Q Tunneling

IN THIS SECTION

- Problem | 1878
- Solution | 1878

Problem

Description

If you create a firewall filter that includes a match condition of `dot1q-tag` or `dot1q-user-priority` and apply the filter on input to a trunk port that participates in a service VLAN, the match condition does not work if the Q-in-Q EtherType is not 0x8100. (When Q-in-Q tunneling is enabled, trunk interfaces are assumed to be part of the service provider or data center network and therefore participate in service VLANs.)

Solution

This is expected behavior. To set the Q-in-Q EtherType to 0x8100, enter the **set dot1q-tunneling ethertype 0x8100** statement at the `[edit ethernet-switching-options]` hierarchy level. You must also configure the other end of the link to use the same Ethertype.

Egress Firewall Filters with Private VLANs

IN THIS SECTION

- Problem | 1878
- Solution | 1879

Problem

Description

If you apply a firewall filter in the output direction to a primary VLAN, the filter also applies to the secondary VLANs that are members of the primary VLAN when the traffic egresses with the primary VLAN tag or isolated VLAN tag, as listed below:

- Traffic forwarded from a secondary VLAN trunk port to a promiscuous port (trunk or access)
- Traffic forwarded from a secondary VLAN trunk port that carries an isolated VLAN to a PVLAN trunk port.
- Traffic forwarded from a promiscuous port (trunk or access) to a secondary VLAN trunk port
- Traffic forwarded from a PVLAN trunk port. to a secondary VLAN trunk port
- Traffic forwarded from a community port to a promiscuous port (trunk or access)

If you apply a firewall filter in the output direction to a primary VLAN, the filter does *not* apply to traffic that egresses with a community VLAN tag, as listed below:

- Traffic forwarded from a community trunk port to a PVLAN trunk port
- Traffic forwarded from a secondary VLAN trunk port that carries a community VLAN to a PVLAN trunk port
- Traffic forwarded from a promiscuous port (trunk or access) to a community trunk port
- Traffic forwarded from a PVLAN trunk port. to a community trunk port

If you apply a firewall filter in the output direction to a community VLAN, the following behaviors apply:

- The filter is applied to traffic forwarded from a promiscuous port (trunk or access) to a community trunk port (because the traffic egresses with the community VLAN tag).
- The filter is applied to traffic forwarded from a community port to a PVLAN trunk port (because the traffic egresses with the community VLAN tag).
- The filter is *not* applied to traffic forwarded from a community port to a promiscuous port (because the traffic egresses with the primary VLAN tag or untagged).

Solution

These are expected behaviors. They occur only if you apply a firewall filter to a private VLAN in the output direction and do not occur if you apply a firewall filter to a private VLAN in the input direction.

Egress Filtering of L2PT Traffic Not Supported

IN THIS SECTION

- Problem | 1880
- Solution | 1880

Problem

Description

Egress filtering of L2PT traffic is not supported on the QFX3500 switch. That is, if you configure L2PT to tunnel a protocol on an interface, you cannot also use a firewall filter to filter traffic for that protocol on that interface in the output direction. If you commit a configuration for this purpose, the firewall filter is not applied to the L2PT-tunneled traffic.

Solution

This is expected behavior.

Cannot Drop BGP Packets in Certain Circumstances

IN THIS SECTION

- Problem | **1880**
- Solution | **1880**

Problem

Description

BGP packets with a time-to-live (TTL) value greater than 1 cannot be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface. BGP packets with TTL value of 1 or 0 can be discarded using a firewall filter applied to a loopback interface or applied on input to a Layer 3 interface.

Solution

This is expected behavior.

Invalid Statistics for Policer

IN THIS SECTION

- Problem | 1881
- Solution | 1881

Problem

Description

If you apply a single-rate two-color policer in more than 128 terms in a firewall filter, the output of the `show firewall` command displays incorrect data for the policer.

Solution

This is expected behavior.

Policers can Limit Egress Filters

IN THIS SECTION

- Problem | 1881
- Solution | 1882

Problem

Description

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries

for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

Solution

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

RELATED DOCUMENTATION

> *Understanding FIP Snooping, FBF, and MVR Filter Scalability*
>
> Configuring Firewall Filters

CHAPTER 28

Configuring Firewall Filter Accounting and Logging (EX9200 Switches)

IN THIS CHAPTER

- Example: Configuring Logging for a Stateless Firewall Filter Term | 1883
- Use the CLI Editor in Configuration Mode | 1889

Example: Configuring Logging for a Stateless Firewall Filter Term

IN THIS SECTION

- Requirements | 1883
- Overview | 1883
- Configuration | 1884
- Verification | 1888

This example shows how to configure a standard stateless firewall filter to log packet headers.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter that logs and counts ICMP packets that have **192.168.207.222** as either their source or destination.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1884
- Configure the Syslog Messages File for the Firewall Facility | 1884
- Configure the Stateless Firewall Filter | 1885
- Apply the Stateless Firewall Filter to a Logical Interface | 1886
- Confirm and Commit Your Candidate Configuration | 1886

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set system syslog file ICMP_filter firewall info
set system syslog file ICMP_filter archive no-world-readable
set firewall family inet filter icmp_syslog term icmp_match from address 192.168.207.222/32
set firewall family inet filter icmp_syslog term icmp_match from protocol icmp
set firewall family inet filter icmp_syslog term icmp_match then count packets
set firewall family inet filter icmp_syslog term icmp_match then syslog
set firewall family inet filter icmp_syslog term icmp_match then accept
set firewall family inet filter icmp_syslog term default_term then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input icmp_syslog
```

Configure the Syslog Messages File for the Firewall Facility

Step-by-Step Procedure

To configure a syslog messages file for the **firewall** facility:

1. Configure a messages file for all syslog messages generated for the **firewall** facility.

```
user@host# set system syslog file ICMP_filter firewall info
```

2. Restrict permission to the archived **firewall** facility syslog files to the root user and users who have the Junos OS maintenance permission.

```
user@host# set system syslog file ICMP_filter archive no-world-readable
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **icmp_syslog** that logs and counts ICMP packets that have **192.168.207.222** as either their source or destination:

1. Create the stateless firewall filter **icmp_syslog**.

```
[edit]
user@host# edit firewall family inet filter icmp_syslog
```

2. Configure matching on the ICMP protocol and an address.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term icmp_match from address 192.168.207.222/32
user@host# set term icmp_match from protocol icmp
```

3. Count, log,, and accept matching packets.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term icmp_match then count packets
user@host# set term icmp_match then syslog
user@host# set term icmp_match then accept
```

4. Accept all other packets.

```
[edit firewall family inet filter icmp_syslog]
user@host# set term default_term then accept
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input icmp_syslog
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the syslog message file for the **firewall** facility by entering the `show system` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

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

```
syslog {
    file ICMP_filter {
        firewall info;
        archive no-world-readable;
    }
}
```

2. Confirm the configuration of the stateless firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter icmp_syslog {
        term icmp_match {
            from {
                address {
                    192.168.207.222/32;
                }
                protocol icmp;
            }
            then {
                count packets;
                log;
                accept;
            }
        }
        term default_term {
            then accept;
        }
    }
}
```

3. Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
```

```
    unit 0 {
        family inet {
            filter {
                input icmp_syslog;
            }
            address 10.1.2.3/30;
        }
    }
}
```

4. If you are done configuring the device, commit your candidate configuration.

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

Verification

To confirm that the configuration is working properly, enter the `show log filter` command:

```
user@host> show log ICMP_filter
Mar 20 08:03:11 hostname feb FW: ge-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:

```
user@host> show log filename
Mar 20 08:18:45 hostname feb FW: ge-0/1/0.0   A icmp 192.168.207.222
192.168.207.223     0     0 (515 packets)
```

RELATED DOCUMENTATION

- System Logging Overview | 1278
- Firewall Filter Logging Actions | 1282
- System Log Explorer
- System Log Explorer

Use the CLI Editor in Configuration Mode

This topic describes basic commands that you can use to enter configuration mode in the CLI editor. The topic also describes commands that you use to navigate the configuration hierarchy, get help, and commit or revert the changes that you make during the configuration session.

Task	Command/ Statement	Example
Edit Your Configuration		

(Continued)

Task	Command/ Statement	Example

Enter configuration mode.
 	
When you start the CLI, the device is in operational mode. You must explicitly enter configuration mode. When you do, the CLI prompt changes from `user@host>` to `user@host#`, and the hierarchy level appears in square brackets.
	`configure`	`user@host> configure`
		
		`[edit]`
		`user@host#`

Create a statement hierarchy.
 	
You can use the `edit` command to simultaneously create a hierarchy and move to that new level in the hierarchy. You cannot use the `edit` command to change the value of identifiers.
	`edit` *`hierarchy-level value`*	`[edit]`
		`user@host# edit security zones security-zone myzone`
		
		`[edit security zones security-zone myzone]`
		`user@host#`

Create a statement hierarchy, and set identifier values.
 	
The `set` command is like `edit`, except that your current level in the hierarchy does not change.
	`set` *`hierarchy-level value`*	`[edit]`
		`user@host# set security zones security-zone myzone`
		
		`[edit]`
		`user@host#`

Navigate the Hierarchy		

Navigate down to an existing hierarchy level.	`edit` *`hierarchy-level`*	`[edit]`
		`user@host# edit security zones`
		
		`[edit security zones]`
		`user@host#`

(Continued)

Task	Command/ Statement	Example

Navigate up one level in the hierarchy.	`up`	`[edit security zones]`
		`user@host# up`
		
		`[edit security]`
		`user@host#`

Navigate to the top of the hierarchy.	`top`	`[edit security zones]`
		`user@host# top`
		
		`[edit]`
		`user@host#`

Commit or Revert Changes		

Commit your configuration.	`commit`	`[edit]`
		`user@host# commit`
		
		`commit complete`

(Continued)

Task	Command/ Statement	Example

Roll changes back from the current session.
 	
Use the `rollback` command to revert all changes from the current configuration session. When you run the `rollback` command before you exit your session or commit changes, the software loads the most recently committed configuration onto the device. You must enter the `rollback` statement at the `edit` level in the hierarchy.
	`rollback`	`[edit]`
		`user@host# rollback`
		
		`load complete`

Exit Configuration Mode		

Commit the configuration, and exit configuration mode.	`commit and-quit`	`[edit]`
		`user@host# commit and-quit`
		
		`user@host>`

Exit configuration mode without committing your configuration.
 	
You must navigate to the top of the hierarchy using the `up` or `top` commands before you can exit configuration mode.
	`exit`	`[edit]`
		`user@host# exit`
		
		`The configuration has been changed but not committed`
		`Exit with uncommitted changes? [yes,no] (yes)`

Get Help		

(Continued)

Task	Command/ Statement	Example
Display a list of valid options for the current hierarchy level.	?	[edit] user@host# edit security zones ? Possible completions: <[Enter> Execute this command > functional-zone Functional zone > security-zone Security zones | Pipe through a command [edit]

RELATED DOCUMENTATION

Understanding Junos OS CLI Configuration Mode

PART 3

Configuring Traffic Policers

Understanding Traffic Policers | 1895

Configuring Policer Rate Limits and Actions | 1966

Configuring Layer 2 Policers | 1976

Configuring Two-Color and Three-Color Traffic Policers at Layer 3 | 2034

Configuring Logical and Physical Interface Traffic Policers at Layer 3 | 2166

Configuring Policers on Switches | 2197

CHAPTER 29

Understanding Traffic Policers

IN THIS CHAPTER

- Policer Implementation Overview | **1896**
- ARP Policer Overview | **1900**
- Example: Configuring ARP Policer | **1902**
- Understanding the Benefits of Policers and Token Bucket Algorithms | **1906**
- Determining Proper Burst Size for Traffic Policers | **1909**
- Controlling Network Access Using Traffic Policing Overview | **1916**
- Traffic Policer Types | **1922**
- Order of Policer and Firewall Filter Operations | **1926**
- Understanding the Frame Length for Policing Packets | **1926**
- Supported Standards for Policing | **1927**
- Hierarchical Policer Configuration Overview | **1928**
- Understanding Enhanced Hierarchical Policers | **1930**
- Packets-Per-Second (pps)-Based Policer Overview | **1934**
- Guidelines for Applying Traffic Policers | **1934**
- Policer Support for Aggregated Ethernet Interfaces Overview | **1935**
- Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface | **1937**
- Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers | **1946**
- Hierarchical Policers on ACX Series Routers Overview | **1949**
- Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**
- Hierarchical Policer Modes on ACX Series Routers | **1953**
- Processing of Hierarchical Policers on ACX Series Routers | **1958**
- Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**
- Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Policer Implementation Overview

The Juniper Networks® Junos® operating system (Junos OS) supports three types of policers:

- *Single-rate two-color policer* — The most common policer. Single-rate means that there is only a single bandwidth and burst rate referenced in the policer. The two colors associated with this policer are red (nonconforming) and green (conforming).
- *Single-rate three-color policer* — Similar to the single-rate two-color policer with the addition of the color yellow. This type also introduces the *committed information rate* (CIR) and a *committed burst rate* (CBR).
- *Two-rate three-color policer* — Builds off of the single-rate three-color policer by adding a second rate tier. *Two-rate* means there is an upper bandwidth limit and associated burst size as well as a *peak information rate* (PIR) and a *peak burst rate* (PBS).

There are two types of token bucket algorithms that can be used, depending on the type of policer that is applied to network traffic. Single-rate two-color policers use the *single token bucket algorithm* to measure traffic flow conformance to a two-color policer rate limit. Single-rate three-color policers and two-rate three-color policers both use the *dual token bucket algorithm* to measure traffic flow conformance to a three-color policer rate. The main difference between these two token bucket algorithms is that the single token bucket algorithm allows bursts of traffic for short periods, whereas the dual token bucket algorithm allows more sustained bursts of traffic. (The remainder of this topic discusses the single token bucket algorithm.)

To configure a policer, you need to set two parameters:

- Bandwidth limit configured in bps (using the **bandwidth-limit** statement)
- Burst size configured in bytes (using the **burst-size-limit** statement)

> **NOTE**: For single-rate two-color policers only, you can also specify the bandwidth limit as a percentage of either the physical interface port speed or the configured *logical interface* shaping rate by using the **bandwidth-percent** *percentage* statement. You cannot configure a policer to use bandwidth percentage for aggregate, tunnel, or software interfaces.

Use the following command to set the policer conditions:

```
user@router# set firewall policer <policer name> if-exceeding ?
Possible completions:
  <[Enter]>            Execute this command
+ apply-groups         Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
```

```
  bandwidth-limit      Bandwidth limit (8000..100000000000 bits per second)
  bandwidth-percent    Bandwidth limit in percentage (1..100 percent)
  burst-size-limit     Burst size limit (1500..100000000000 bytes)
  |                    Pipe through a command
```

The bandwidth limit parameter is used to determine the average rate limit applied to the traffic, while the burst-size parameter is used to allow for short periods of traffic bursting (back-to-back traffic at average rates that exceed the configured bandwidth limit). Once you apply a set of policer configuration settings (bandwidth limit and burst size), the configured values are adjusted to hardware programmable values. The conversion adjustment introduced is normally less than 1 percent of the configured bandwidth limit. This adjustment is needed because the software allows you to configure the bandwidth limit and burst size to any value within the specified ranges, but those values must be adjusted to the nearest value that can be programmed in the hardware.

The policer bandwidth limit configuration in the hardware is represented by two values: the *credit update frequency* and the *credit size*. The credit update frequency is used by the hardware to determine how frequently tokens (bits of unused bandwidth) are added to the token bucket. The credit size is based on the number of tokens that can fit in the token bucket. The MX Series, M120, M320 routers, and EX Series switches contain a set of credit update frequencies instead of having a single credit update frequency to minimize the adjustment difference from the configured bandwidth limit and to support a wide range of policer bandwidth rates (from 40 Kbps to 40 Gbps). One of the frequencies is used to program the policer (bandwidth limit and burst size) in the hardware.

The burst size is based on the overall traffic load and allows bursts of traffic to exceed the configured bandwidth limit. A policer with a large burst size effectively disables the configured bandwidth limit function, so the burst size must be relative to the configured bandwidth limit. You need to consider the traffic patterns in your network before determining the burst size. For more information about determining burst size, see "Determining Proper Burst Size for Traffic Policers" on page 1909.

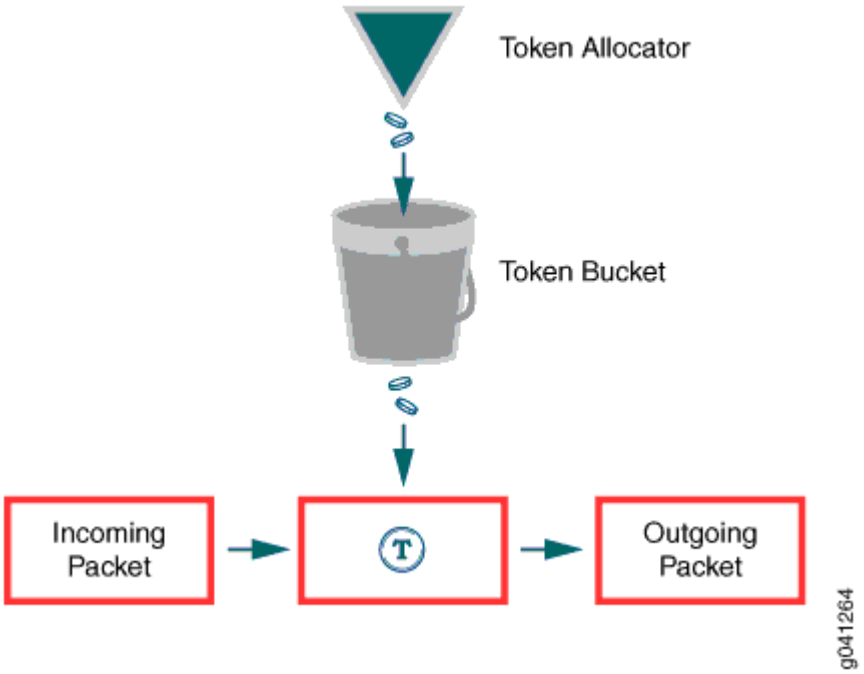
The configured burst size is adjusted in the hardware to a value that is based on the configured bandwidth limit. The burst size extends the configured bandwidth limit for bursty traffic that exceeds the configured bandwidth limit.

When a policer is applied to the traffic at an interface, the initial capacity for traffic bursting is equal to the number of bytes specified in the **burst-size-limit** statement.

Figure 76 on page 1898 represents how a policer is implemented using the token bucket algorithm. The token allocator allocates tokens to the policer based on the configured bandwidth limit, which is the token size multiplied by the token arrival rate.

token size x token arrival rate = policer rate (configured bandwidth limit)

Figure 76: Token Bucket Algorithm

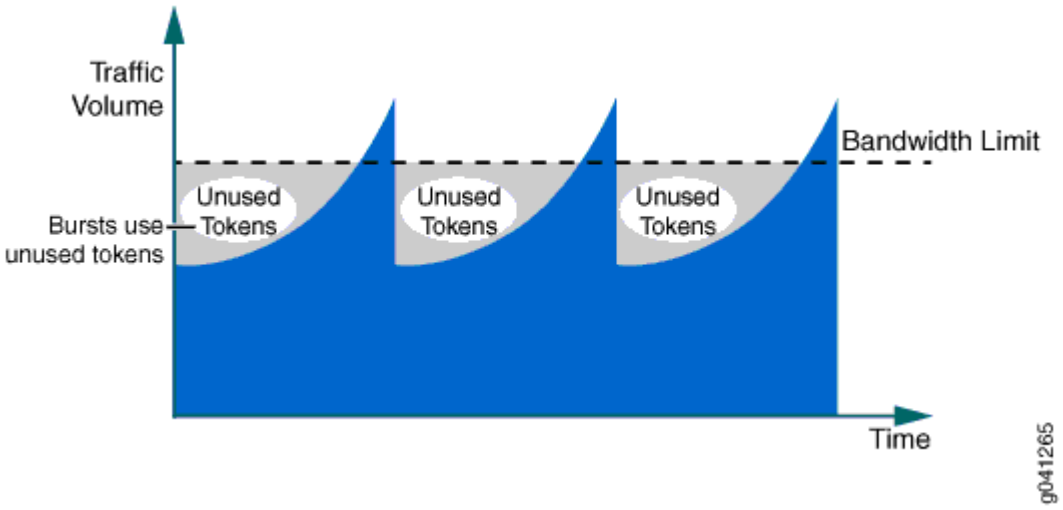


When a packet arrives at an interface configured with a policer, tokens that represent the number of bits that correspond to the length of the packet are used (or “cashed in”) from the token bucket. If the token arrival rate is higher than the rate of traffic so that there are tokens not being used, the token bucket is filled to capacity, and arriving tokens “overflow” the bucket and are lost. The token bucket depth represents the single user-configured burst size for the policer.

If there are tokens in the token bucket and the incoming traffic rate is higher than the token rate (the configured policer rate, bandwidth limit), the traffic can use the tokens until the bucket is empty. The token consumption rate can be as high as the incoming traffic rate, which creates the burst of traffic shown in Figure 77 on page 1899.

By using the token bucket algorithm, the average bandwidth rate being allowed is close to the configured bandwidth limit while simultaneously supporting bursty traffic, as shown in Figure 77 on page 1899.

Figure 77: Traffic Behavior Using Policer and Burst Size



NOTE: The measured length of a packet changes according to the family type that the policer applies to. If the policer is applied under the `family inet` hierarchy, the policer considers only the IPv4 packet length. If the policer is applied under the `family vpls` hierarchy, the entire Ethernet frame (including the Ethernet MAC header) is included in the packet length.

The major factor that affects the policer shaping result is not the conversion adjustment, but the traffic pattern since most network traffic is not consistent and is not sent at a constant rate. Due to the fluctuation of the incoming traffic rate, some of the allocated tokens are not used. As a result, the shaped traffic rate is lower than you might expect, and the TCP connection behavior discussed in "Understanding the Benefits of Policers and Token Bucket Algorithms" on page 1906 is a typical example of this. To alleviate this effect of the lower shaped traffic rate, a proper burst size configuration is required.

RELATED DOCUMENTATION

Understanding the Benefits of Policers and Token Bucket Algorithms | **1906**

Determining Proper Burst Size for Traffic Policers | **1909**

ARP Policer Overview

IN THIS SECTION

- Benefits of the ARP Policer | **1901**

Sending IP packets on a multi access network requires mapping from an IP address to a media access control (MAC) address (the physical or hardware address).

In an Ethernet environment, Address Resolution Protocol (ARP) is used to map a MAC address to an IP address. ARP dynamically binds the IP address (the logical address) to the correct MAC address. Before sending unicast IP packets, ARP discovers the MAC address used for the Ethernet interface where the IP address is configured.

Hosts that use ARP maintain a cache of discovered Internet-to-Ethernet address mappings to minimize the number of ARP broadcast messages. To keep the cache from growing too large, an entry is removed if it is not used within a certain period of time. Before sending a packet, the host searches its cache for Internet-to-Ethernet address mapping. If you cannot find the mapping, the host sends an ARP request.

Starting in Junos OS Release 18.4R1, you can apply policers on ARP traffic on SRX Series Firewalls. Support for ARP policers on pseudowire interfaces on MX Series routers is available in Junos OS Release 20.2R1. The configuration principles are the same.

You can configure rate limiting for the policer by specifying the bandwidth and the burst-size limit. Packets exceeding the policer limits are discarded. The traffic to the Routing Engine is controlled by applying the policer on ARP traffic. Using policers helps prevent network congestion caused by broadcast storms. You can use policers to specify rate limits on traffic. A firewall filter configured with a policer permits only traffic within a specified set of rate limits, thereby providing protection from denial-of-service (DoS) attacks. Traffic that exceeds the rate limits specified by the policer is either discarded immediately or is marked as lower priority than traffic that is within the rate limits. The switch discards the lower-priority traffic when there is traffic congestion.

A policer applies two types of rate limits on 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 an algorithm to enforce a limit on average bandwidth while allowing bursts up to a specified maximum value. You can define specific classes of traffic on an interface and apply a set of rate

limits to each class. After you name and configure a policer, it is stored as a template. You can then use the policer in a firewall filter configuration.

NOTE: On SRX5400, SRX5600, and SRX5800 devices, ARP policer actions are applied on the SPUs as well as on the Routing Engine. For example, SPU A handles 15000 packets of ARP traffic, and SPU B handles 5000 packets. A policer is configured as rate-limit 10K, discard and applied to the ARP protocol. As a result, SPU A discards 5000 packets of ARP traffic and forwards 10000 packets to the Routing Engine, and SPU B forwards 5000 packets of ARP the Routing Engine. The Routing Engine therefore receives a total of 15000 packets of ARP traffic.

Benefits of the ARP Policer

- Prevents network congestion caused by broadcast storms
- Protects Routing Engines on SRX Series Firewalls that are impacted by broadcast storms
- Provides protection from denial-of-service (DoS) attacks

Release History Table

Release	Description
20.2R1	Support for ARP policers on pseudowire interfaces on MX Series routers is available in Junos OS Release 20.2R1.
18.4R1	Starting in Junos OS Release 18.4R1, you can apply policers on ARP traffic on SRX Series Firewalls.

RELATED DOCUMENTATION

bandwidth-limit (Hierarchical Policer)

burst-size-limit (Hierarchical Policer)

Example: Configuring ARP Policer | **1902**

Example: Configuring ARP Policer

IN THIS SECTION

- Requirements | 1902
- Overview | 1902
- Configuration | 1903
- Verification | 1905

This example shows how to configure an Address Resolution Protocol (ARP) policer on SRX Series Firewalls.

Support for ARP policers on pseudowire interfaces on MX Series routers is available in Junos OS Release 20.2R1. The configuration principles are the same as shown here.

Requirements

This example uses the following hardware and software components:

- SRX Series Firewall.
- Junos OS Release 18.4R1 or later.

Before you begin, see "ARP Policer Overview" on page 1900.

Overview

ARP is used to map a MAC address to an IP address. ARP dynamically binds the IP address (the logical address) to the correct MAC address. Before IP unicast packets can be sent, ARP discovers the MAC address used by the Ethernet interface where the IP address is configured. This feature is supported on all SRX Series Firewalls. The traffic to the Routing Engine on the SRX Series Firewall is controlled by applying the policer on ARP. This prevents network congestion caused by broadcast storms.

NOTE: A default ARP policer named *__default_arp_policer__* is used and shared by all Ethernet interfaces with `family inet` configured, by default.

On MX Series routers, you can create policers for ARP traffic on pseudowire interfaces. (You configure rate limiting for the policer by specifying the bandwidth and the burst-size limit of a firewall policer and attaching the policy to a pseudowire interface, just like you would any other interface, and apply the

ARP policer to a pseudowire interface at the `[edit interfaces interface-name unit unit-number family inet policer arp policy-name]` level of the hierarchy. Traffic that exceeds the specified rate limits can be dropped or marked as low priority and delivered when congestion permits.

Configuration

IN THIS SECTION

- Configuring ARP Policer on Interface | **1903**

This example shows how to configure rate limiting for the policer by specifying the bandwidth and the burst-size limit.

Configuring ARP Policer on Interface

CLI Quick Configuration

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

```
set firewall policer arp_limit if-exceeding bandwidth-limit 1m
set firewall policer arp_limit if-exceeding burst-size-limit 1m
set firewall policer arp_limit then discard
set interfaces ge-0/0/7 unit 0 family inet policer arp arp_limit
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Use the CLI Editor in Configuration Mode" on page 1889 in the CLI User Guide.

To configure the ARP policer:

1. Specify the name of the policer.

```
[edit firewall]
user@host# set policer arp-limit
```

2. Configure rate limiting for the policer.

   - Specify the bandwidth limit in bits per second (bps) to control the traffic rate on an interface:

     ```
     [edit firewall policer arp_limit]
     user@host# set if-exceeding bandwidth-limit 1m
     ```

     The range for the bandwidth limit is 1 through 150,000 bps.

   - Specify the burst-size limit (the maximum allowed burst size in bytes) to control the amount of traffic bursting:

     ```
     [edit firewall policer arp_limit]
     user@host# set if-exceeding burst-size-limit 1m
     ```

     To determine the value for the burst-size limit, multiply the bandwidth of the interface on which the filter is applied by the amount of time to allow a burst of traffic at that bandwidth to occur:

     burst size = (bandwidth) * (allowable time for burst traffic)

     The range for the burst-size limit is 1 through 150,00 bytes.

3. Specify the policer action discard to discard packets that exceed the rate limits.

```
[edit firewall]
user@host# set policer arp_limit then discard
```

   Discard is the only supported policer action.

4. Configure the interfaces.

```
user@host# set interfaces ge-0/0/7 unit 0 family inet policer arp arp_limit
```

Results

From configuration mode, confirm your configuration by entering the `show firewall` command. If the output does not display the intended configuration, repeat the instructions in this example to correct.

```
[edit]
    user@host# show firewall
    policer arp_limit {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 1m;
    }
    then discard;
}
[edit]
    user@host# show interfaces
    ge-0/0/7 {
    unit 0 {
        family inet {
            policer {
                arp arp_limit;
            }
        }
    }
}
```

After you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying the results of arp policer | 1906

To confirm that the configuration is working properly, perform these tasks:

Verifying the results of arp policer

Purpose

Verify the results of the Arp policer.

Action

From the top of the configuration in operational mode, enter the `show policer` *`policer-name`* command.

```
user@host> show policer arp_limit-ge-0/0/7.0-inet-arp
Policers:
Name                                                Bytes              Packets
arp_limit-ge-0/0/7.0-inet-arp                           0                    0
```

Meaning

The `show policer` *`policer-name`* command displays the names of all firewall filters and policers that are configured on the device.

Release History Table

Release	Description
20.2R1	Support for MX Series routers is available in Junos OS Release 20.2R1, and the configuration principles are the same as shown here.

RELATED DOCUMENTATION

ARP Policer Overview | 1900

Understanding the Benefits of Policers and Token Bucket Algorithms

IN THIS SECTION

Scenario 1: Single TCP Connection | 1907

Scenario 2: Multiple TCP Connections | 1908

This topic describes some scenarios that demonstrate how difficult it is to control traffic that comes into your network without the help of policers and the token bucket algorithm. These scenarios assume that traffic is coming from a TCP-based connection. Depending on the number of TCP connections, policers can have different affects on rate limits.

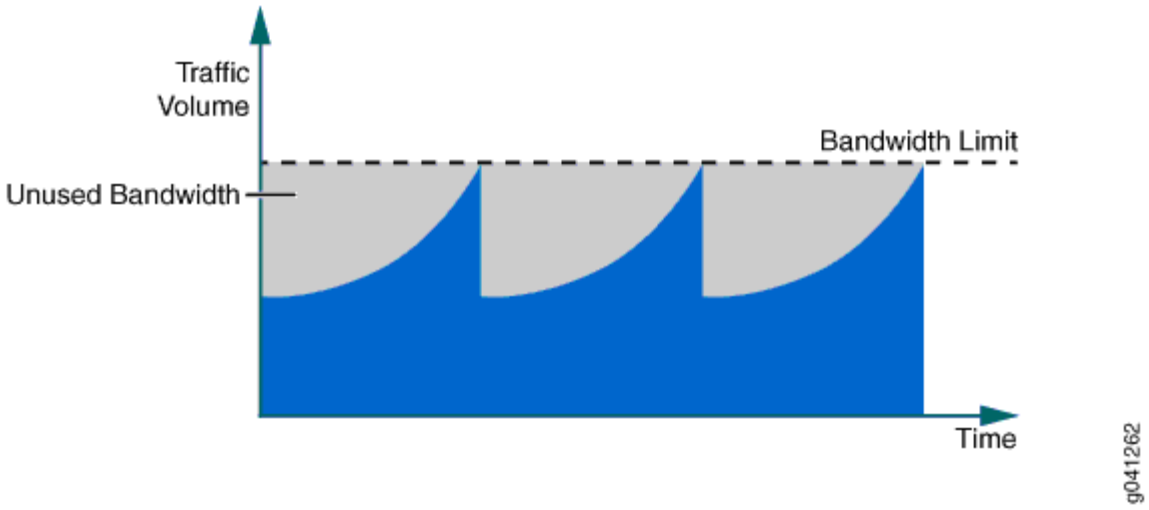
This topic presents the following scenarios:

- "Scenario 1: Single TCP Connection" on page 1907
- "Scenario 2: Multiple TCP Connections" on page 1908

Scenario 1: Single TCP Connection

Figure 78 on page 1907 shows the traffic loading on an interface with a policer configured. When the traffic rate reaches the configured bandwidth limit (which results in a packet drop), a TCP slow-start mechanism reduces the traffic rate down to half of what it was. When the traffic rate rises again, the same cycle repeats.

Figure 78: Policer Behavior with a Single TCP Connection



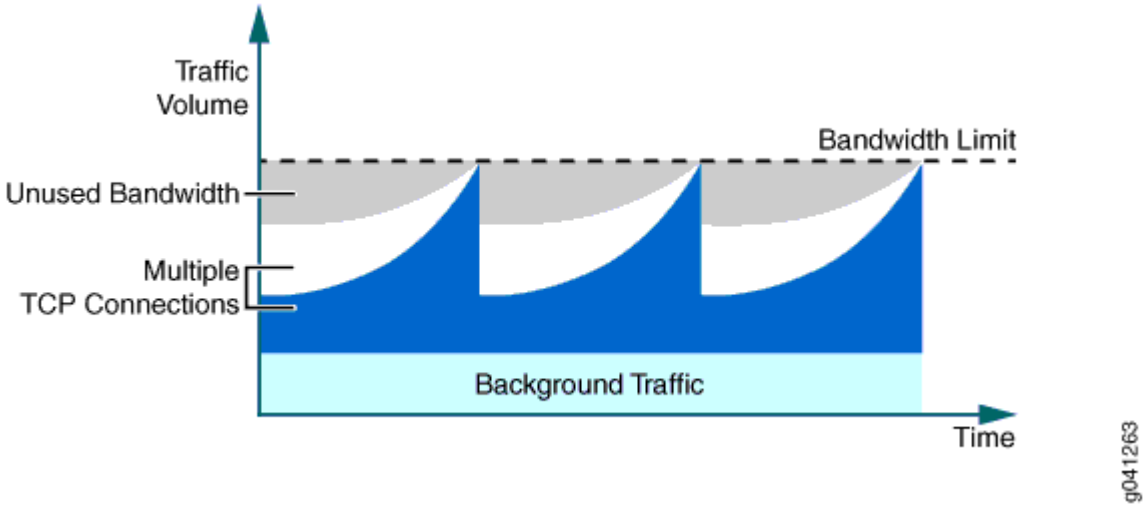
The problem presented in this scenario is that some bandwidth is available, but it is not being used by the traffic. The unused bandwidth shown in Figure 78 on page 1907 is the result of an overall data

throughput that is lower than the configured bandwidth value. This example is an extreme case because there is only a single TCP connection.

Scenario 2: Multiple TCP Connections

With multiple TCP connections or some background non-TCP-based traffic, there is less unused bandwidth, as depicted in Figure 79 on page 1908. However, the same issue of unused bandwidth still exists if all the TCP connections experience a drop when the aggregated traffic rate exceeds the configured bandwidth limit.

Figure 79: Policer Behavior with Background Traffic (Multiple TCP Connections)



To reduce the problem of unused bandwidth in your network, you can configure a burst size.

RELATED DOCUMENTATION

Policer Implementation Overview | **1896**

Determining Proper Burst Size for Traffic Policers | **1909**

Determining Proper Burst Size for Traffic Policers

IN THIS SECTION

- Policer Burst Size Limit Overview | 1909
- Effect of Burst-Size Limit | 1910
- Two Methods for Calculating Burst-Size Limit | 1911
- Comparison of the Two Methods | 1912

Policer Burst Size Limit Overview

A policer burst-size limit controls the number of bytes of traffic that can pass unrestricted through a policed interface when a burst of traffic pushes the average transmit or receive rate above the configured bandwidth limit. The actual number of bytes of bursty traffic allowed to pass through a policed interface can vary from zero to the configured burst-size limit, depending on the overall traffic load.

By configuring a proper burst size, the effect of a lower shaped rate is alleviated. Use the `burst-size-limit` statement to configure the burst size.

NOTE: If you set the burst-size limit too low, too many packets will be subjected to rate limiting. If you set the burst-size limit too high, too few packets will be rate-limited.

Consider these two main factors when determining the burst size to use:

- The allowed duration of a blast of traffic on the line.
- The burst size is large enough to handle the maximum transmission unit (MTU) size of the packets.

The following general guidelines apply to choosing a policer burst-size limit:

- A burst-size limit should not be set lower than 10 times the MTU of the traffic on the interface to be policed.
- The amount of time to allow a burst of traffic at the full line rate of a policed interface should not be lower than 5 ms.
- The minimum and maximum values you can specify for a policer burst-size limit depends on the policer type (two-color or three-color).

BEST PRACTICE: The preferred method for choosing a burst-size limit is based on the line rate of the interface on which you apply the policer and the amount of time you want to allow a burst of traffic at the full line rate.

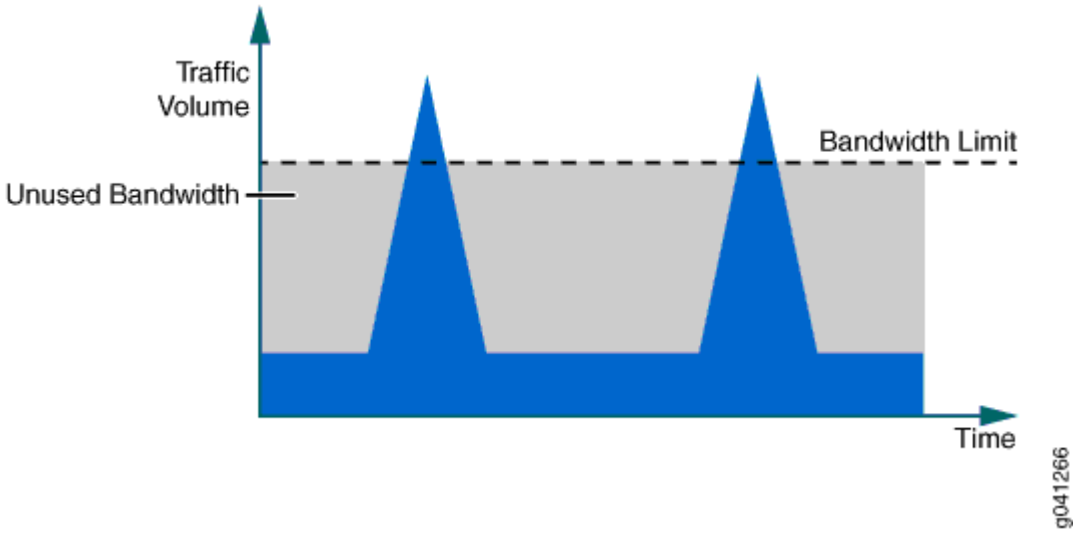
Effect of Burst-Size Limit

Bursty traffic requires a relatively large burst size so that extra tokens can be allocated into the token bucket for upcoming traffic to use.

Bursty Traffic Policed Without a Burst-Size Limit

Figure 80 on page 1910 shows an extreme case of bursty traffic where the opportunity to allocate tokens is missed, and the bandwidth goes unused because a large burst size is not configured.

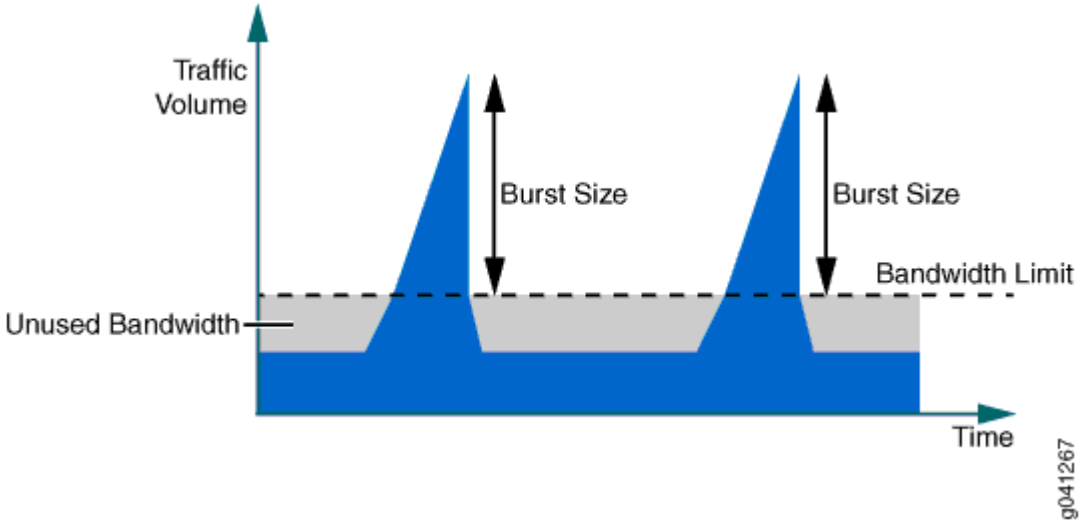
Figure 80: Bursty Traffic Without Configured Burst Size (Excessive Unused Bandwidth)



Burst-Size Limit Configured to Match Bandwidth Limit and Flow Burstiness

Figure 81 on page 1911 depicts how bandwidth usage changes when a large burst size is configured to handle bursty traffic. The large burst size minimizes the amount of unused bandwidth because tokens are being allocated in between the bursts of traffic that can be used during traffic peaks. The burst size determines the depth of the token bucket.

Figure 81: Bursty Traffic with Configured Burst Size (Less Unused Bandwidth)



Burst-Size Limit That Depletes All Accumulated Tokens

Configuring a large burst size for the unused tokens creates another issue. If the burst size is set to a very large value, the burst of traffic can be transmitted from the interface at line rate until all the accumulated tokens in the token bucket are used up. This means that configuring a large burst size can allow too many packets to avoid rate limiting, which can lead to a traffic rate that exceeds the bandwidth limit for an extended period of time.

If the average rate is considered within 1 second, the rate is still below the configured bandwidth limit. However, the downstream device might not be able to handle bursty traffic, so some packets might be dropped.

Two Methods for Calculating Burst-Size Limit

For policers configured on MX Series, M120, and M320 routers, and EX Series switches, configurable burst-size limit values range from 1 ms through 600 ms of traffic at the policer rate (the configured bandwidth limit).

Because one burst size is not suitable for every traffic pattern, select the best burst size for an interface by performing experimental configurations. For your first test configuration, select the burst-size limit by using one of the calculation methods described in the next two sections.

Calculation Based on Interface Bandwidth and Allowable Burst Time

If the bandwidth of the policed interface is known, the preferred method for calculating the policer burst-size limit is based on the following values:

- `bandwidth`—Line rate of the policed interface (in bps units)
- `burst-period`—Allowable traffic-burst time (5 ms or longer)

To calculate policer bandwidth in bytes:

```
bandwidth  X  burst-period  /  8
```

Calculation Based on Interface Traffic MTU

If the bandwidth of the policed interface is unknown, calculate the policer burst-size limit based on the following value:

- `interface MTU`—Maximum transmission unit (in bytes) for the policed interface.

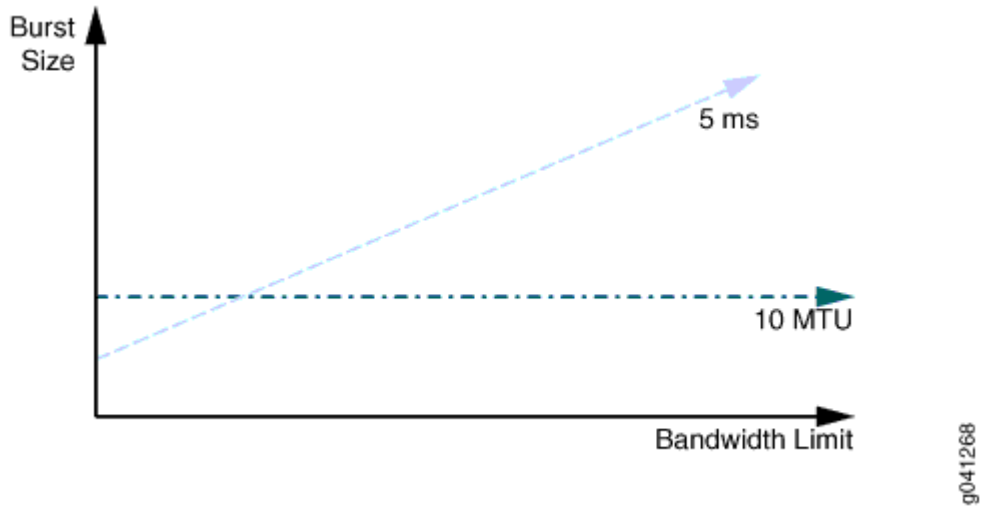
To calculate policer bandwidth in bytes:

```
interface MTU  X  10
```

Comparison of the Two Methods

Figure 82 on page 1913 illustrates the relationship between the policer rate (the configured bandwidth limit) and the effective burst-size limit for the two methods of calculating the best policer burst-size limit. For the method based on interface bandwidth and allowable burst time, the correlation is labeled **5 ms**. For the method based on MTU size, the correlation is labeled **10 MTU**.

Figure 82: Comparing Burst Size Calculation Methods



For a policer burst-size limit calculated using the **5 ms** method, the effective burst-size limit is proportional to the configured bandwidth limit. With a very low bandwidth limit, the effective burst-size limit might be so small that the policer rate-limits traffic more aggressively than desired. For example, a traffic “burst” consisting of two MTU-sized packets might be rate-limited. In this scenario, a policer burst-size limit calculated using the **10 MTU** method appears to be a better choice.

10 x MTU Method for Selecting Initial Burst Size for Gigabit Ethernet with 100 Kbps Bandwidth

The following sequence illustrates the use of the 10 x MTU method for selecting an initial burst size for test configurations for a Gigabit Ethernet interface configured with a 100 Kbps bandwidth limit:

1. If you configure a 100 ms burst-size limit, the maximum amount of traffic allowed to pass through the interface unrestricted is 1250 bytes, calculated as follows:

$$100\ \text{Kbps} \times 100\ \text{ms} = \frac{100{,}000\ \text{bps} \times 0.1\ \text{s}}{8\ \text{bits per byte}} = 1250\ \text{bytes}$$

2. In theory, a 10 x MTU burst size would allow up to 15,000 bytes to pass unrestricted. However, the maximum configurable burst-size limit for MX Series, M120, and M320 routers is 600 ms of the bandwidth limit. If you configure the maximum burst-size limit of 600 ms of the bandwidth limit, the

maximum amount of traffic allowed to pass through the interface unrestricted is 7500 bytes, calculated as follows:

```
                      100,000 bps x 0.6 s
100 Kbps x 600 ms  =  ———————————————————  =  7500 bytes
                        8 bits per byte
```

On a Gigabit Ethernet interface, a configured burst-size limit of 600 ms creates a burst duration of 60 µs at Gigabit Ethernet line rate, calculated as follows:

```
 7500 bytes          60,000 bits
————————————  =  ——————————————————  =  0.00006 s  =  60 µs
   1 Gbps         1,000,000,000 bps
```

3. If the downstream device is unable to handle the amount of bursty traffic allowed using the initial burst size configuration, reduce the burst-size limit until you achieve acceptable results.

5 ms Method for Selecting Initial Burst Size for Gigabit Ethernet Interface with 200 Mbps Bandwidth

The following sequence illustrates the use of the 5 ms method for selecting an initial burst size for test configurations for a Gigabit Ethernet interface configured with a 200 Mbps bandwidth limit. This example calculation shows how a larger burst-size limit can affect the measured bandwidth rate.

1. If you configure a 5 ms burst-size limit, the maximum amount of traffic allowed to pass through the interface unrestricted is 125,000 bytes (approximately 83 1500-byte packets), calculated as follows:

```
                     200,000,000 bps x 0.005 s
200 Mbps x 5 ms  =  ———————————————————————  =  125,000 bytes
                       8 bits per byte
```

On a Gigabit Ethernet interface, a configured burst-size limit of 5 ms creates a burst duration of 1 ms at Gigabit Ethernet line rate, calculated as follows:

```
 125,000 bytes         1,000,000 bits
———————————————  =  ———————————————————  =  0.001 s  =  1 ms
      1 Gbps         1,000,000,000 bps
```

The average bandwidth rate in 1 second becomes 200 Mbps + 1 Mbps = 201 Mbps, which is a minimal increase over the configured bandwidth limit at 200 Mbps.

2. If you configure a 600 ms burst-size limit, the maximum amount of traffic allowed to pass through the interface unrestricted is 15 Mbytes (approximately 10,000 1500-byte packets), calculated as follows:

```
                      200,000,000 bps x 0.6 s
200 Mbps x 600 ms  =  ———————————————————————  =  15,000,000 bytes
                        8 bits per byte
```

On a Gigabit Ethernet interface, a configured burst-size limit of 600 ms creates a burst duration of 120 ms at Gigabit Ethernet line rate, calculated as follows:

```
15,000,000 bytes     120,000,000 bits
————————————————  =  ———————————————————  =  0.12 s  =  120 ms
     1 Gbps         1,000,000,000 bps
```

The average bandwidth rate in 1 second becomes 200 Mbps + 120 Mbps = 320 Mbps, which is much higher than the configured bandwidth limit at 200 Mbps.

200 Mbps Bandwidth Limit, 5 ms Burst Duration

If a 200 Mbps bandwidth limit is configured with a 5 ms burst size, the calculation becomes **200 Mbps x 5 ms = 125 Kbytes**, which is approximately 83 1500-byte packets. If the 200 Mbps bandwidth limit is

configured on a Gigabit Ethernet interface, the burst duration is **125000 bytes / 1 Gbps = 1 ms** at the Gigabit Ethernet line rate.

200 Mbps Bandwidth Limit, 600 ms Burst Duration

If a large burst size is configured at 600 ms with the bandwidth limit configured at 200 Mbps, the calculation becomes **200 Mbps x 600 ms = 15 Mbytes**. This creates a burst duration of 120 ms at the Gigabit Ethernet line rate. The average bandwidth rate in 1 second becomes **200 Mbps + 15 Mbytes = 320 Mbps**, which is much higher than the configured bandwidth limit at 200 Mbps. This example shows that a larger burst size can affect the measured bandwidth rate.

RELATED DOCUMENTATION

Policer Implementation Overview | **1896**

Understanding the Benefits of Policers and Token Bucket Algorithms | **1906**

Controlling Network Access Using Traffic Policing Overview

IN THIS SECTION

- Congestion Management for IP Traffic Flows | **1916**
- Traffic Limits | **1917**
- Traffic Color Marking | **1918**
- Forwarding Classes and PLP Levels | **1920**
- Policer Application to Traffic | **1921**

Congestion Management for IP Traffic Flows

Traffic policing, also known as *rate limiting*, is an essential component of network access security that is designed to thwart denial-of-service (DoS) attacks. Traffic policing enables you to control the maximum rate of IP traffic sent or received on an interface and also to partition network traffic into multiple priority levels, also known as *classes of service*. A policer defines a set of traffic rate limits and sets consequences for traffic that does not conform to the configured limits. Packets in a traffic flow that do not conform to traffic limits are either discarded or marked with a different forwarding class or packet loss priority (PLP) level.

With the exception of policers configured to rate-limit aggregate traffic (all protocol families and logical interfaces configured on a physical interface), you can apply a policer to all IP packets in a Layer 2 or Layer 3 traffic flow at a *logical interface*.

With the exception of policers configured to rate-limit based on physical interface media rate, you can apply a policer to specific IP packets in a Layer 3 traffic flow at a logical interface by using a stateless *firewall filter*.

You can apply a policer to inbound or outbound interface traffic. Policers applied to inbound traffic help to conserve resources by dropping traffic that does not need to be routed through a network. Dropping inbound traffic also helps to thwart denial-of-service (DoS) attacks. Policers applied to outbound traffic control the bandwidth used.

NOTE: Traffic policers are instantiated on a per-PIC basis. Traffic policing does not work when the traffic for one local policy decision function (L-PDF) subscriber is distributed over multiple Multiservices PICs in an AMS group.

Traffic Limits

Junos OS policers use a *token bucket algorithm* to enforce a limit on an average transmit or receive rate of traffic at an interface while allowing bursts of traffic up to a maximum value based on the configured bandwidth limit and configured burst size. The token bucket algorithm offers more flexibility than a *leaky bucket algorithm* in that you can allow a specified traffic burst before starting to discard packets or apply a penalty such as packet output-queuing priority or packet-drop priority.

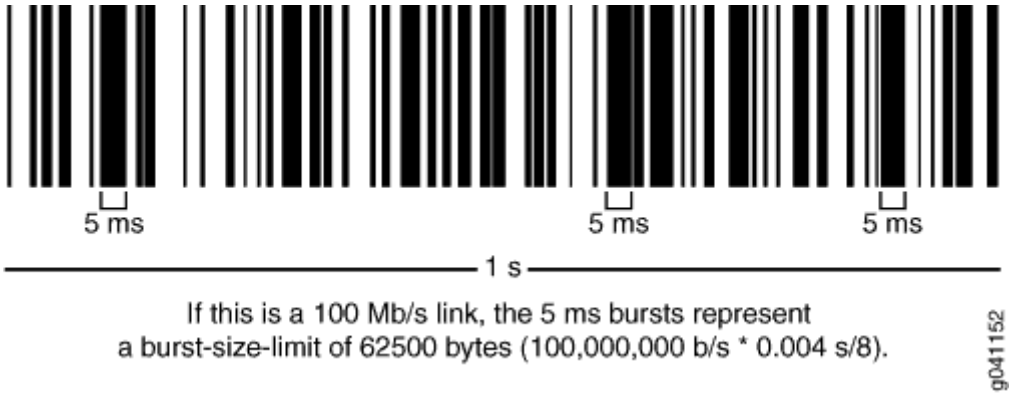
In the token-bucket model, the bucket represents the rate-limiting function of the policer. Tokens are added to the bucket at a fixed rate, but once the specified depth of the bucket is reached, tokens allocated after cannot be stored and used. Each token represents a “credit” for some number of bits, and tokens in the bucket are “cashed in” for the ability to transmit or receive traffic at the interface. When sufficient tokens are present in the bucket, a traffic flow continues unrestricted. Otherwise, packets might be dropped or else re-marked with a lower forwarding class, a higher packet loss priority (PLP) level, or both.

- The rate at which tokens are added to the bucket represents the highest average transmit or receive rate in bits per second allowed for a given service level. You specify this highest average traffic rate as the *bandwidth limit* of the policer. If the traffic arrival rate (or fixed bits-per-second) is so high that at some point insufficient tokens are present in the bucket, then the traffic flow is no longer conforming to the traffic limit. During periods of relatively low traffic (traffic that arrives at or departs from the interface at average rates below the token arrival rate), unused tokens accumulate in the bucket.

- The depth of the bucket in bytes controls the amount of back-to-back bursting allowed. You specify this factor as the *burst-size limit* of the policer. This second limit affects the average transmit or

receive rate by limiting the number of bytes permitted in a transmission burst for a given interval of time. Bursts exceeding the current burst-size limit are dropped until there are sufficient tokens available to permit the burst to proceed.

Figure 83: Network Traffic and Burst Rates



As shown in the figure above, a UPC bar code is a good facsimile of what traffic looks like on the line; an interface is either transmitting (bursting at full rate) or it is not. The black lines represent periods of data transmission and the white space represents periods of silence when the token bucket can replenish.

Depending on the type of policer used, packets in a policed traffic flow that surpasses the defined limits might be implicitly set to a higher PLP level, assigned to a configured forwarding class or set to a configured PLP level (or both), or simply discarded. If packets encounter downstream congestion, packets with a `low` PLP level are less likely to be discarded than those with a `medium-low`, `medium-high`, or `high` PLP level.

Traffic Color Marking

Based on the particular set of traffic limits configured, a policer identifies a traffic flow as belonging to one of either two or three categories that are similar to the colors of a traffic light used to control automobile traffic.

- *Single-rate two-color*—A two-color marking policer (or “policer” when used without qualification) meters the traffic stream and classifies packets into two categories of packet loss priority (PLP) according to a configured bandwidth and burst-size limit. You can mark packets that exceed the bandwidth and burst-size limit in some way, or simply discard them.

  A policer is most useful for metering traffic at the port (physical interface) level.

- *Single-rate three-color*—This type of policer is defined in RFC 2697, *A Single Rate Three Color Marker*, as part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a

Differentiated Services (DiffServ) environment. This type of policer meters traffic based on the configured committed information rate (CIR), committed burst size (CBS), and the excess burst size (EBS). Traffic is marked as belonging to one of three categories (green, yellow, or red) based on whether the packets arriving are below the CBS (green), exceed the CBS (yellow) but not the EBS, or exceed the EBS (red).

A single-rate three-color policer is most useful when a service is structured according to packet length and not peak arrival rate.

- *Two-rate three-color*—This type of policer is defined in RFC 2698, *A Two Rate Three Color Marker*, as part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a Differentiated Services (DiffServ) environment. This type of policer meters traffic based on the configured CIR and peak information rate (PIR), along with their associated burst sizes, the CBS and *peak burst size* (PBS). Traffic is marked as belonging to one of three categories (green, yellow, or red) based on whether the packets arriving are below the CIR (green), exceed the CIR (yellow) but not the PIR, or exceed the PIR (red).

  A two-rate three-color policer is most useful when a service is structured according to arrival rates and not necessarily packet length.

Policer actions are implicit or explicit and vary by policer type. The term *Implicit* means that Junos assigns the loss-priority automatically. Table 109 on page 1919 describes the policer actions.

Table 109: Policer Actions

Policer	Marking	Implicit Action	Configurable Action
Single-rate two-color	Green (Conforming)	Assign low loss priority	None
	Red (Nonconforming)	None	Assign low or high loss priority, assign a forwarding class, or discard On some platforms, you can assign medium-low or medium-high loss priority
Single-rate three-color	Green (Conforming)	Assign low loss priority	None
	Yellow (Above the CIR and CBS)	Assign medium-high loss priority	None

Table 109: Policer Actions *(Continued)*

Policer	Marking	Implicit Action	Configurable Action
	Red (Above the EBS)	Assign high loss priority	Discard
Two-rate three-color	Green (Conforming)	Assign low loss priority	None
	Yellow (Above the CIR and CBS)	Assign medium-high loss priority	None
	Red (Above the PIR and PBS)	Assign high loss priority	Discard

Forwarding Classes and PLP Levels

A packet's forwarding class assignment and PLP level are used by the Junos OS class of service (CoS) features. The Junos OS CoS features include a set of mechanisms that you can use to provide differentiated services when best-effort traffic delivery is insufficient. For router (and switch) interfaces that carry IPv4, IPv6, and MPLS traffic, you can configure CoS features to take in a single flow of traffic entering at the edge of your network and provide different levels of service across the network—internal forwarding and scheduling (queuing) for output—based on the forwarding class assignments and PLP levels of the individual packets.

NOTE: Forwarding-class or loss-priority assignments performed by a policer or a stateless firewall filter override any such assignments performed on the ingress by the CoS default IP precedence classification at all logical interfaces or by any configured behavior aggregate (BA) classifier that is explicitly mapped to a logical interface.

Based on CoS configurations, packets of a given forwarding class are transmitted through a specific output queue, and each output queue is associated with a transmission service level defined in a *scheduler*.

Based on other CoS configurations, when packets in an output queue encounter congestion, packets with higher loss-priority values are more likely to be dropped by the random early detection (RED) algorithm. Packet loss priority values affect the scheduling of a packet without affecting the packet's relative ordering within the traffic flow.

Policer Application to Traffic

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 want to use it. This eliminates the need to define the same policer values more than once.

You can apply a policer to a traffic flow in either of two ways:

- You can configure a standard stateless firewall filter that specifies the `policer` *`policer-name`* nonterminating action or the `three-color-policer (single-rate | two-rate)` *`policer-name`* nonterminating action. When you apply the standard filter to the input or output at a logical interface, the policer is applied to all packets of the filter-specific protocol family that match the conditions specified in the filter configuration.

  With this method of applying a policer, you can define specific classes of traffic on an interface and apply traffic rate-limiting to each class.

- You can apply a policer directly to an interface so that traffic rate-limiting applies to all traffic on that interface, regardless of protocol family or any match conditions.

You can configure policers at the queue, logical interface, or Layer 2 (MAC) level. Only a single policer is applied to a packet at the egress queue, and the search for policers occurs in this order:

- Queue level
- Logical interface level
- Layer 2 (MAC) level

RELATED DOCUMENTATION

Stateless Firewall Filter Overview | **776**

Traffic Policer Types | **1922**

Order of Policer and Firewall Filter Operations | **1926**

Packet Flow Through the Junos OS CoS Process Overview

Traffic Policer Types

IN THIS SECTION

- Single-Rate Two-Color Policers | **1922**
- Three-Color Policers | **1923**
- Hierarchical Policers | **1924**
- Two-Color and Three-Color Policer Options | **1924**

Single-Rate Two-Color Policers

You can use a single-rate two-color policer, or “policer” when used without qualification, to rate-limit a traffic flow to an average bits-per-second arrival rate (specified by the single specified bandwidth limit) while allowing bursts of traffic for short periods (controlled by the single specified burst-size limit). This type of policer categorizes a traffic flow as either green (conforming) or red (nonconforming). Packets in a green flow are implicitly set to a `low` loss priority and then transmitted. Packets in a red flow are handled according to actions specified in the policer configuration. Packets in a red flow can be marked —set to a specified forwarding class, set to a specified loss priority, or both—or they can be discarded.

A single-rate two-color policer is most useful for metering traffic at the port (physical interface) level.

Basic Single-Rate Two-Color Policer

You can apply a basic single-rate two-color policer to Layer 3 traffic in either of two ways: as an interface policer or as a *firewall filter* policer. You can apply the policer as an <!--<new-term>interface policer</new-term>-->, meaning that you apply the policer directly to a *logical interface* at the protocol family level. If you want to apply the policer to selected packets only, you can apply the policer as a *firewall filter policer*, meaning that you reference the policer in a stateless firewall filter term and then apply the filter to a logical interface at the protocol family level.

Bandwidth Policer

A bandwidth policer is simply a single-rate two-color policer that is defined using a bandwidth limit specified as a percentage value rather than as an absolute number of bits per second. When you apply the policer (as an interface policer or as a firewall filter policer) to a logical interface at the protocol family level, the effective bandwidth limit is calculated based on either the physical interface media rate or the logical interface configured shaping rate.

Logical Bandwidth Policer

A logical bandwidth policer is a bandwidth policer for which the effective bandwidth limit is calculated based on the logical interface configured shaping rate. You can apply the policer as a firewall filter policer only, and the firewall filter must be configured as an interface-specific filter. When you apply an interface-specific filter to multiple logical interfaces on supported routing platforms, any `count` or `policer` actions act on the traffic stream entering or exiting each individual interface, regardless of the sum of traffic on the multiple interfaces.

Three-Color Policers

The Junos OS supports two types of three-color policers: single-rate and two-rate. The main difference between a single-rate and a two-rate policer is that the single-rate policer allows bursts of traffic for short periods, while the two-rate policer allows more sustained bursts of traffic. Single-rate policing is implemented using a single token-bucket model, so that periods of relatively low traffic must occur between traffic bursts to allow the token bucket to refill. Two-rate policing is implemented using a dual token-bucket model, which allows bursts of traffic for longer periods.

Single-Rate Three-Color Policers

The single-rate three-color type of policer is defined in RFC 2697, *A Single Rate Three Color Marker*. You use this type of policer to rate-limit a traffic flow to a single rate and three traffic categories (green, yellow, and red). A single-rate three-color policer defines a *committed* bandwidth limit and burst-size limit plus an *excess* burst-size limit. Traffic that conforms to the committed traffic limits is categorized as green (conforming). Traffic that conforms to the bandwidth limit while allowing bursts of traffic as controlled by the excess burst-size limit is categorized as yellow. All other traffic is categorized as red.

A single-rate three-color policer is most useful when a service is structured according to packet length, not peak arrival rate.

Two-Rate Three-Color Policers

The two-rate three-color type of policer is defined in RFC 2698, *A Two Rate Three Color Marker*. You use this type of policer to rate-limit a traffic flow to two rates and three traffic categories (green, yellow, and red). A two-rate three-color policer defines a *committed* bandwidth limit and burst-size limit plus a *peak* bandwidth limit and burst-size limit. Traffic that conforms to the committed traffic limits is categorized as green (conforming). Traffic that exceeds the committed traffic limits but remains below the peak traffic limits is categorized as yellow. Traffic that exceeds the peak traffic limits is categorized as red.

A two-rate three-color policer is most useful when a service is structured according to arrival rates and not necessarily packet length.

Hierarchical Policers

You can use a hierarchical policer to rate-limit ingress Layer 2 traffic at a physical or logical interface and apply different policing actions based on whether the packets are classified for expedited forwarding (EF) or for a lower priority output queue. This feature is supported on SONET interfaces hosted on M40e, M120, and M320 edge routers with incoming Flexible PIC Concentrators (FPCs) as SFPC and outgoing FPCs as FFPC, and on T320, T640, and T1600 core routers with Enhanced Intelligent Queuing (IQE) PICs.

Two-Color and Three-Color Policer Options

Both two-color and three-color policers can be configured with the following options:

Logical Interface (Aggregate) Policers

A logical interface policer—also called an aggregate policer—is a two-color or three-color policer that you can apply to multiple protocol families on the same logical interface without creating multiple instances of the policer. You apply a logical interface policer directly to a logical interface configuration (and not by referencing the policer in a stateless firewall filter and then applying the filter to the logical interface).

- You can apply the policer at the interface logical unit level to rate-limit all traffic types, regardless of the protocol family.

  When applied in this manner, the logical interface policer will be used by all traffic types (inet, intet6, etc.) and across all layers (layer 2, layer 3) no matter where the policer is attached on the logical interface.

- You can also apply the policer at the logical interface protocol family level, to rate-limit traffic for a specific protocol family.

You can apply a logical interface policer to unicast traffic only. For information about configuring a stateless firewall filter for flooded traffic, see "*Applying Forwarding Table Filters*" in the "Traffic Sampling, Forwarding, and Monitoring" section of the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

Physical Interface Policers

A physical interface policer is a two-color or three-color policer that applies to all logical interfaces and protocol families configured on a physical interface, even if the logical interfaces belong to different routing instances. You apply a physical interface policer to a logical interface at the protocol level through a physical interface filter only, but rate limiting is performed aggregately for all logical interfaces and protocol families configured on the underlying physical interface.

This feature enables you to use a single policer instance to perform aggregate policing for different protocol families and different logical interfaces on the same physical interface.

Policers Applied to Layer 2 Traffic

In addition to hierarchical policing, you can also apply single-rate two-color policers and three-color policers (both single-rate and two-rate) to Layer 2 input or output traffic. You must configure the two-color or three-color policer as a logical interface policer and reference the policer in the interface configuration at the logical unit level, and not at the protocol level. You cannot apply a two-color or three-color policer to Layer 2 traffic as a stateless firewall filter action.

Multifield Classification

Like behavior aggregate (BA) classification, which is sometimes referred to as class-of-service (CoS) value traffic classification, multifield classification is a method of classifying incoming traffic by associating each packet with a forwarding class, a packet loss priority level, or both. The CoS scheduling configuration assigns packets to output queues based on forwarding class. The CoS random early detection (RED) process uses the drop probability configuration, output queue fullness percentage, and packet loss priority to drop packets as needed to control congestion at the output stage.

BA classification and multifield classification use different fields of a packet to perform traffic classification. BA classification is based on a *CoS value* in the IP packet header. Multifield classification can be based on *multiple fields* in the IP packet header, including CoS values. Multifield classification is used instead of BA classification when you need to classify packets based on information in the packet other than the CoS values only. Multifield classification is configured using a stateless firewall filter term that matches on any packet header fields and associates matched packets with a forwarding class, a loss priority, or both. The forwarding class or loss priority can be set by a firewall filter action or by a policer referenced as a firewall filter action.

RELATED DOCUMENTATION

- Controlling Network Access Using Traffic Policing Overview | **1916**
- Order of Policer and Firewall Filter Operations | **1926**
- Two-Color Policer Configuration Overview | **2034**
- Three-Color Policer Configuration Overview | **2118**
- Hierarchical Policer Configuration Overview | **1928**
- Two-Color Policing at Layer 2 Overview
- Three-Color Policing at Layer 2 Overview

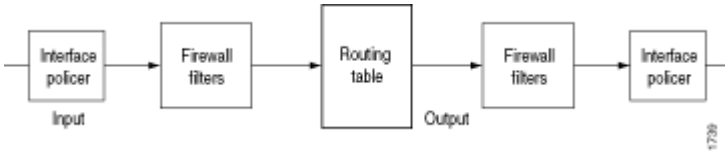
Order of Policer and Firewall Filter Operations

You can apply a both a traffic policer and a stateless *firewall filter* (with or without policing actions) to a single *logical interface* at the same time. In this case, the order of precedence of operations is such that policers applied directly to the logical interface are evaluated before input filters but after output filters.

- If an input firewall filter is configured on the same logical interface as a policer, the policer is executed first.
- If an output firewall filter is configured on the same logical interface as a policer, the firewall filter is executed first.

Figure 84 on page 1926 illustrates the order of policer and firewall filter processing at the same interface.

Figure 84: Incoming and Outgoing Policers and Firewall Filters



NOTE: Order of policer and firewall operations applies only to policers that have been applied directly to the logical interface and not to policers referenced within a firewall filter.

RELATED DOCUMENTATION

How Standard Firewall Filters Evaluate Packets | **793**

Comparison of Routing Policies and Firewall Filters | **9**

Two-Color Policer Configuration Overview | **2034**

Three-Color Policer Configuration Overview | **2118**

Hierarchical Policer Configuration Overview | **1928**

Understanding the Frame Length for Policing Packets

Table 110 on page 1927 describes the packet lengths that are considered when you use a traffic policer.

Table 110: Packet Lengths Considered for Traffic Policers

Protocol	Policing Packet Lengths
Any	L3 frame including header
IPv4	L3 frame including header
IPv6	L3 frame including header
MPLS	L3 frame including header
VPLS	MAC
Bridge	MAC
CCC	MAC

RELATED DOCUMENTATION

Policer Overhead to Account for Rate Shaping in the Traffic Manager | **2106**

Supported Standards for Policing

Three-color policers are part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a Differentiated Services (DiffServ) environment, which is described and defined in the following RFCs:

- RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*
- RFC 2475, *An Architecture for Differentiated Service*
- RFC 2597, *Assured Forwarding PHB Group*
- RFC 2598, *An Expedited Forwarding PHB*
- RFC 2698, *A Two Rate Three Color Marker*

In a DiffServ environment, the most significant 6 bits of the type-of-service (ToS) octet in the IP header contain a value called the *Differentiated Services code point* (DSCP). Within the DSCP field, the most significant 3 bits are interpreted as the *IP precedence* field, which can be used to select different per-hop forwarding treatments for the packet.

Hierarchical Policer Configuration Overview

Hierarchically rate-limits Layer 2 ingress traffic for all protocol families. Cannot be applied to egress traffic, Layer 3 traffic, or at a specific protocol level of the interface hierarchy.

Supported on the following interfaces:

- SONET interfaces hosted on M40e, M120, and M320 edge routers with incoming FPCs as SFPC and outgoing FPCs as FFPC.
- SONET interfaces hosted on T320, T640, and T1600 core routers with Enhanced Intelligent Queuing (IQE) PICs.
- Ethernet interfaces on Gigabit Ethernet Intelligent Queuing 2 (IQ2) and Ethernet Enhanced IQ2 (IQ2E) PICs.
- MX Series routers with MPC or DPC.

Table 111 on page 1928 describes the hierarchy levels at which you can configure and apply hierarchical policers.

Table 111: Hierarchical Policer Configuration and Application Summary

Policer Configuration	Layer 2 Application	Key Points
Hierarchical Policer		

Table 111: Hierarchical Policer Configuration and Application Summary *(Continued)*

Policer Configuration	Layer 2 Application	Key Points

Aggregate and premium policing components of a hierarchical policer:
 	
`[edit firewall]`
`hierarchical-policer` *`policer-name`*
`{`
`  aggregate {`
`  if-exceeding {`
`  bandwidth-limit` *`bps`*`;`
`  burst-size-limit` *`bytes`*`;`
`  }`
`  then {`
`  discard;`
`  forwarding-class` *`class-name`*`;`
`  loss-priority` *`supported-value`*`;`
`  }`
`  }`
`  premium {`
`  if-exceeding {`
`  bandwidth-limit` *`bps`*`;`
`  burst-size-limit` *`bytes`*`;`
`  }`
`  then {`
`  discard;`
`  }`
`  }`
`}`
	Option A—Apply directly to Layer 2 input traffic on a physical interface:
	
	`[edit interfaces]`
	`interface-name` `{`
	`  layer2-policer {`
	`  input-hierarchical-policer` *`policer-name`*`;`
	`  }`
	`}`
		Hierarchically rate-limit Layer 2 ingress traffic for all protocol families and logical interfaces configured on a physical interface.
		
		Include the `layer2-policer` configuration statement at the `[edit interfaces` *`interface-name`*`]` hierarchy level.
		
		NOTE: If you apply a hierarchical policer at a physical interface, you cannot also apply a hierarchical policer to any of the member logical interfaces.

	Option B—Apply directly to Layer 2 input traffic on a logical interface.
	
	`[edit interfaces]`
	`interface-name` `{`
	`  unit` *`unit-number`* `{`
	`  layer2-policer {`
	`  input-hierarchical-policer` *`policer-name`*`;`
	`  }`
	`  }`
	`}`
		Hierarchically rate-limit Layer 2 ingress traffic for all protocol families configured on a specific logical interface.
		
		Include the `layer2-policer` configuration statement at the `[edit interfaces` *`interface-name`* `unit` *`unit-number`*`]` hierarchy level.
		
		NOTE: You must configure at least one protocol family for the logical interface.

RELATED DOCUMENTATION

Hierarchical Policers | **1976**

Understanding Enhanced Hierarchical Policers

IN THIS SECTION

- Guidelines for configuring enhanced hierarchical policer | **1931**
- Example: Configuring an enhanced hierarchical policer | **1931**

Use the enhanced hierarchical policer configuration to rate limit traffic based on packets classified on the traffic priority. Configure traffic policing at four levels of hierarchies with respect to the traffic priority.

NOTE: This feature is available only on ACX7100-32C, ACX7100-48L, ACX7509, and ACX7024 devices.

In an enhanced hierarchical policer configuration, up to four policers are defined. Each policer maps to a traffic priority. The four traffic priorities, arranged as per their order of precedence are High, Medium-High, Medium-Low, and Low. The traffic priorities are hierarchical – High is the traffic priority with the highest precedence and Low is the traffic priority with the lowest precedence. It implies that a policer defined for the High traffic priority has a higher precedence than the rest of the policers or a policer defined for the Low traffic priority has a lower precedence than the rest of the policers.

All policers (one or up to four) in an enhanced hierarchical policer configuration, consume bandwidth from a maximum allotted bandwidth – in Table 1 this maximum allotted bandwidth is 65 mbps. Each policer is allotted a Confirmed Information Rate (CIR) and Maximum Confirmed Information Rate from this maximum allotted bandwidth. As a guideline, the CIR and Max CIR values are always the same for the policer with the highest precedence.

Residue bandwidth or unused bandwidth is carried over to lower precedence policers. As can be noted in Table 112 on page 1931, medium-high policer inherits unused bandwidth from high policer. Medium-low policer inherits from high and medium-high policers. Low policer inherits from the other three higher precedence policers. It is recommended that MAX CIR of a particular level is equal to the CIR of current level + combined CIR of previous/top levels.

Table 112: Example enhanced hierarchical policer configuration

Policer Configurations		
Policer-level/traffic-priority	**CIR**	**MAX CIR**
high	5mbps	5mbps
medium-high	10mbps	15mbps
medium-low	20mbps	35mbps
low	30mbps	65mbps

Guidelines for configuring enhanced hierarchical policer

- An enhanced hierarchical policer is filter-specific. Filter-specific policer semantics is to be used for hierarchical policer as multiple terms will point to same policer.
- Counter name must be same for all the terms mapped to enhanced-hierarchical-policer action under same hierarchy level.
- It is mandatory to configure all the levels of an enhanced hierarchical policer with respective policer bandwidth rates and burst size configurations. If there is no requirement to configure all four levels, the unwanted levels must specify least supported CIR, MAX CIR and CBS rates. It is recommended that firewall filter terms not be mapped to these unwanted levels.
- Each enhanced hierarchical policer level must be configured with the action to discard the packets exceeding the configured bandwidth.

Example: Configuring an enhanced hierarchical policer

In this example:

- An enhanced hierarchical policer is defined.
- A firewall filter is defined and policer is applied in the firewall filter. The firewall filter is applied to an interface.
- Policer statistics is displayed.

Requirements:

- Junos OS Release 23.3 R1 or later.
- An ACX7100-32C, ACX7100-48L, ACX7509, or ACX7024 device.

Step-by-Step Procedure

1. Define the policer name.

```
[edit]
user@host# set firewall enhanced-hierarchical-policer hpol
```

2. Define committed information rate (CIR), maximum committed information rate (MIR), and committed burst size (CBS) for the four traffic priorities, namely high, medium-high, medium-low, and low.

```
user@host# set firewall enhanced-hierarchical-policer hpol filter-specific
user@host# set firewall enhanced-hierarchical-policer hpol high committed-information-rate 5m
user@host# set firewall enhanced-hierarchical-policer hpol high max-committed-information-
rate 5m
user@host# set firewall enhanced-hierarchical-policer hpol high committed-burst-size 5k
user@host# set firewall enhanced-hierarchical-policer hpol high then discard
user@host# set firewall enhanced-hierarchical-policer hpol medium-high committed-information-
rate 10m
user@host# set firewall enhanced-hierarchical-policer hpol medium-high max-committed-
information-rate 15m
user@host# set firewall enhanced-hierarchical-policer hpol medium-high committed-burst-size
15k
user@host# set firewall enhanced-hierarchical-policer hpol medium-high then discard
user@host# set firewall enhanced-hierarchical-policer hpol medium-low committed-information-
rate 20m
user@host# set firewall enhanced-hierarchical-policer hpol medium-low max-committed-
information-rate 35m
user@host# set firewall enhanced-hierarchical-policer hpol medium-low committed-burst-size 35k
user@host# set firewall enhanced-hierarchical-policer hpol medium-low then discard
user@host# set firewall enhanced-hierarchical-policer hpol low committed-information-rate 30m
user@host# set firewall enhanced-hierarchical-policer hpol low max-committed-information-rate
65m
```

```
user@host# set firewall enhanced-hierarchical-policer hpol low committed-burst-size 65k
user@host# set firewall enhanced-hierarchical-policer hpol low then discard
```

3. Define a firewall filter. Apply the enhanced hierarchical policer by specifying the traffic priority in the action of the firewall filter term.

```
user@host# set firewall family inet filter hpol-inet interface-specific
user@host# set firewall family inet filter hpol-inet term platinum from dscp af11
user@host# set firewall family inet filter hpol-inet term platinum then enhanced-hierarchical-
policer hpol traffic-priority high
user@host# set firewall family inet filter hpol-inet term gold from dscp af12
user@host# set firewall family inet filter hpol-inet term gold then enhanced-hierarchical-
policer hpol traffic-priority medium-high
user@host# set firewall family inet filter hpol-inet term silver from dscp af13
user@host# set firewall family inet filter hpol-inet term silver then enhanced-hierarchical-
policer hpol traffic-priority medium-low
user@host# set firewall family inet filter hpol-inet term dflt then enhanced-hierarchical-
policer hpol traffic-priority low
```

4. Apply the firewall filter to an interface.

```
user@host# set interfaces et-0/0/1 unit 0 family inet address 100.1.1.6/32
user@host# set interfaces et-0/0/1 unit 0 family inet filter input hpol-inet
```

5. Display enhanced hierarchical policer statistics. The dropped/red bytes and packets are displayed per level.

```
user@host# show firewall
Filter: hpol-inet-et-0/0/1.0-i
Enchanced Hierarchical Policers:
Name           Bytes     Packets
hpol
  High            0          0
  Medium-High     0          0
  Medium-Low      0          0
  Low             0          0
```

Packets-Per-Second (pps)-Based Policer Overview

In a modern network environment, both denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks are very common. Over time, these attacks have evolved from brute force types of attacks, where the attacker might try to overrun a connection's available bandwidth with a vast amount of directed traffic to more low-and-slow attacks that use smaller packets, sent at a slower rate to target specific resources in order to deny service.

Traffic policers, both interface-based and filter-based, have been available to mitigate brute force types of DDoS attacks since Junos OS Release 9.6. These policers operate by limiting the traffic rate through a logical interface or by limiting the traffic rate as the "nonterminating action" on page 871 within a firewall filter.

In Junos OS Release 15.1 and earlier releases, there were two parameters available for policers: bandwidth and burst-size. The unit of measure for the bandwidth parameter is bits per second (bps), and the unit of measure for the burst-size parameter is bytes (B). See "Policer Bandwidth and Burst-Size Limits" on page 1966 for details. Policers defined within these parameters are not effective at stopping low-and-slow types of DDoS attacks.

Starting in Junos OS Release 16.1, traffic policers can be defined using packets per second (pps) with the `pps-limit` and `packet-burst` statements. The unit of measure for `pps-limit` is packets per second (pps), and the unit of measure for `packet-burst` is packets. These pps-based policers are more effective at mitigating low-and-slow types of DDoS attacks.

Policers configured with the `if-exceeding-pps` statement are applied in the same manner and in the same locations as bandwidth-based policers. Pps-based policers cannot be applied simultaneously with bandwidth-based policers. Only one policer can be applied at a time except for hierarchical policers, which allow the configuration of two policing actions based on traffic classification.

RELATED DOCUMENTATION

- *if-exceeding-pps*
- *pps-limit*
- *packet-burst*

Guidelines for Applying Traffic Policers

The following general guidelines pertain to applying traffic policers:

- Only one type of policer can be applied to the input or output of the same physical or logical interface. For example, you are not allowed to apply a policer and a hierarchical policer in the same direction at the same logical interface.
- Chaining of policers—that is, applying policers to both a port and the logical interfaces of that port—is not allowed.
- A maximum of 64 policers is supported per physical or logical interface, provided no behavior aggregate (BA) classification—traffic classification based on CoS values in the packet headers—is applied to the logical interface.
- The policer should be independent of BA classification. Without BA classification, all traffic on an interface is treated either as expedited forwarding (EF) or non-EF, based on the configuration. With BA classification, a physical or logical interface can support up to 64 policers. The interface might be a physical interface or logical interface.
- With BA classification, the miscellaneous traffic (the traffic *not* matching any of the BA classification DSCP/EXP bits) is policed as non-EF traffic. No separate policers are installed for this traffic.
- Policers can be applied to unicast packets only. For information about configuring a filter for flooded traffic, see *Applying Forwarding Table Filters*.

RELATED DOCUMENTATION

Two-Color Policer Configuration Overview | **2034**

Three-Color Policer Configuration Overview | **2118**

Hierarchical Policer Configuration Overview | **1928**

Policer Support for Aggregated Ethernet Interfaces Overview

Aggregated interfaces support single-rate policers, three-color marking policers, two-rate three-color marking policers, hierarchical policers, and percentage-based policers. By default, policer bandwidth and burst-size applied on aggregated bundles is not matched to the user-configured bandwidth and burst-size.

You can configure interface-specific policers applied on an aggregated Ethernet bundle or an aggregated SONET bundle to match the effective bandwidth and burst-size to user-configured values. The `shared-bandwidth-policer` statement is required to achieve this match behavior.

This capability applies to all interface-specific policers of the following types: single-rate policers, single-rate three-color marking policers, two-rate three-color marking policers, and hierarchical policers.

Percentage-based policers match the bandwidth to the user-configured values by default, and do not require shared-bandwidth-policer configuration. The `shared-bandwidth-policer` statement causes a split in burst-size for percentage-based policers.

NOTE: This feature is supported on the following platforms: T Series routers (excluding T4000 Type 5 FPCs), M120, M10i, M7i (CFEB-E only), M320 (SFPC only), MX240, MX480, and MX960 with DPC, MIC, and MPC interfaces, and EX Series switches.

The following usage scenarios are supported:

- Interface policers used by the following configuration:

```
[edit] interfaces (aeX | asX) unit unit-num family  family policer [input | output | arp]
```

- Policers and three-color policers (both single-rate three-color marking and two-rate three-color marking) used inside interface-specific filters; that is, filters that have an interface-specific keyword and are used by the following configuration:

```
[edit] interfaces (aeX | asX) unit unit-num family family filter [input | output]
```

- Common-edge service filters, which are derived from CLI-configured filters and thus inherit interface-specific properties. All policers and three-color policers used by these filters are also affected.

The following usage scenarios are not supported:

- Policers and three-color policers used inside filters that are not interface specific; such a filter is meant to be shared across multiple interfaces.
- Any implicit policers or policers that are part of implicit filters; for example, the default ARP policer applied to an aggregate Ethernet interface. Such a policer is meant to be shared across multiple interfaces.
- Prefix-specific action policers.

To configure this feature, include the `shared-bandwidth-policer` statement at the following hierarchy levels: `[edit firewall policer policer-name]`, `[edit firewall three-color-policer policer-name]`, or `[edit firewall hierarchical-policer policer-name]`.

RELATED DOCUMENTATION

shared-bandwidth-policer

Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface

IN THIS SECTION

- Requirements | 1937
- Overview | 1937
- Configuration | 1938
- Verification | 1944

This example shows how to configure a single-rate two-color policer as a physical interface policer.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 1938

A *physical interface policer* specifies rate-limiting for aggregate traffic, which encompasses all protocol families and logical interfaces configured on a physical interface, even if the interfaces belong to different routing instances.

You can apply a physical interface policer to Layer 3 input or output traffic only by referencing the policer from a stateless firewall filter that is configured for specific a specific protocol family (not for `family any`) and configured as a physical interface filter. You configure the filter terms with match

conditions that select the types of packets you want to rate-limit, and you specify the physical interface policer as the action to apply to matched packets.

NOTE: Physical interface policers/filters are not supported for list filters.

Topology

The physical interface policer in this example, `shared-policer-A`, rate-limits to 10,000,000 bps and permits a maximum burst of traffic of 500,000 bytes. You configure the policer to discard packets in nonconforming flows, but you could instead configure the policer to re-mark nonconforming traffic with a forwarding class, a packet loss priority (PLP) level, or both.

To be able to use the policer to rate-limit IPv4 traffic, you reference the policer from an IPv4 physical interface filter. For this example, you configure the filter to pass the policer IPv4 packets that meet either of the following match terms:

- Packets received through TCP and with the IP precedence fields `critical-ecp` (0xa0), `immediate` (0x40), or `priority` (0x20)
- Packets received through TCP and with the IP precedence fields `internet-control` (0xc0) or `routine` (0x00)

You could also reference the policer from physical interface filters for other protocol families.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1939**
- Configuring the Logical Interfaces on the Physical Interface | **1939**
- Configuring a Physical Interface Policer | **1940**
- Configuring an IPv4 Physical Interface Filter | **1941**
- Applying the IPv4 Physical interface Filter to Reference the Physical Interface Policers | **1943**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces so-1/0/0 unit 0 family inet address 192.168.1.1/24
set interfaces so-1/0/0 unit 0 family vpls
set interfaces so-1/0/0 unit 1 family mpls
set firewall policer shared-policer-A physical-interface-policer
set firewall policer shared-policer-A if-exceeding bandwidth-limit 100m burst-size-limit 500k
set firewall policer shared-policer-A then discard
set firewall family inet filter ipv4-filter physical-interface-filter
set firewall family inet filter ipv4-filter term tcp-police-1 from precedence [ critical-ecp
immediate priority ]
set firewall family inet filter ipv4-filter term tcp-police-1 from protocol tcp
set firewall family inet filter ipv4-filter term tcp-police-1 then policer shared-policer-A
set firewall family inet filter ipv4-filter term tcp-police-2 from precedence [ internet-control
routine ]
set firewall family inet filter ipv4-filter term tcp-police-2 from protocol tcp
set firewall family inet filter ipv4-filter term tcp-police-2 then policer shared-policer-A
set interfaces so-1/0/0 unit 0 family inet filter input ipv4-filter
```

Configuring the Logical Interfaces on the Physical Interface

Step-by-Step Procedure

To configure the logical interfaces on the physical interface:

1. Enable configuration of logical interfaces.

```
[edit]
user@host# edit interfaces so-1/0/0
```

2. Configure protocol families on logical unit 0.

```
[edit interfaces so-1/0/0]
user@host# set unit 0 family inet address 192.168.1.1/24
user@host# set unit 0 family vpls
```

3. Configure protocol families on logical unit 1.

```
[edit interfaces so-1/0/0]
user@host# set unit 1 family mpls
```

Results

Confirm the configuration of the firewall filter by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
        family inet {
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

Configuring a Physical Interface Policer

Step-by-Step Procedure

To configure a physical interface policer:

1. Enable configuration of the two-color policer.

```
[edit]
user@host# edit firewall policer shared-policer-A
```

2. Configure the type of two-color policer.

```
[edit firewall policer shared-policer-A]
user@host# set physical-interface-policer
```

3. Configure the traffic limits and the action for packets in a nonconforming traffic flow.

```
[edit firewall policer shared-policer-A]
user@host# set if-exceeding bandwidth-limit 100m burst-size-limit 500k
user@host# set then discard
```

For a physical interface filter, the actions you can configure for packets in a nonconforming traffic flow are to discard the packets, assign a forwarding class, assign a PLP value, or assign both a forwarding class and a PLP value.

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer shared-policer-A {
    physical-interface-policer;
    if-exceeding {
        bandwidth-limit 100m;
        burst-size-limit 500k;
    }
    then discard;
}
```

Configuring an IPv4 Physical Interface Filter

Step-by-Step Procedure

To configure a physical interface policer as the action for terms in an IPv4 physical interface policer:

1. Configure a standard stateless firewall filter under a specific protocol family.

```
[edit]
user@host# edit firewall family inet filter ipv4-filter
```

   You cannot configure a physical interface firewall filter for `family any`.

2. Configure the filter as a physical interface filter so that you can apply the physical interface policer as an action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set physical-interface-filter
```

3. Configure the first term to match IPv4 packets received through TCP with the IP precedence fields `critical-ecp`, `immediate`, or `priority` and to apply the physical interface policer as a filter action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set term tcp-police-1 from precedence [ critical-ecp immediate priority ]
user@host# set term tcp-police-1 from protocol tcp
user@host# set term tcp-police-1 then policer shared-policer-A
```

4. Configure the first term to match IPv4 packets received through TCP with the IP precedence fields `internet-control` or `routine` and to apply the physical interface policer as a filter action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set term tcp-police-2 from precedence [ internet-control routine ]
user@host# set term tcp-police-2 from protocol tcp
user@host# set term tcp-police-2 then policer shared-policer-A
```

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
```

```
    filter ipv4-filter {
        physical-interface-filter;
        term tcp-police-1 {
            from {
                precedence [ critical-ecp immediate priority ];
                protocol tcp;
            }
            then policer shared-policer-A;
        }
        term tcp-police-2 {
            from {
                precedence [ internet-control routine ];
                protocol tcp;
            }
            then policer shared-policer-A;
        }
    }
}
policer shared-policer-A {
    physical-interface-policer;
    if-exceeding {
        bandwidth-limit 100m;
        burst-size-limit 500k;
    }
    then discard;
}
```

Applying the IPv4 Physical interface Filter to Reference the Physical Interface Policers

Step-by-Step Procedure

To apply the physical interface filter so it references the physical interface policers:

1. Enable configuration of IPv4 on the logical interface.

```
[edit]
user@host# edit interfaces so-1/0/0 unit 0 family inet
```

2. Apply the IPv4 physical interface filter in the input direction.

```
[edit interfaces so-1/0/0 unit 0 family inet]
user@host# set filter input ipv4-filter
```

Results

Confirm the configuration of the firewall filter by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
        family inet {
            filter {
                input ipv4-filter;
            }
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to an Interface | **1945**
- Displaying the Number of Packets Processed by the Policer at the Logical Interface | **1945**

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to an Interface

Purpose

Verify that the firewall filter `ipv4-filter` is applied to the IPv4 input traffic at logical interface `so-1/0/0.0`.

Action

Use the `show interfaces statistics` operational mode command for logical interface `so-1/0/0.0`, and include the `detail` option. In the **Protocol inet** section of the command output, the **Input Filters** field shows that the firewall filter `ipv4-filter` is applied in the input direction.

```
user@host> show interfaces statistics so-1/0/0 detail
  Logical interface so-1/0/0.0 (Index 79) (SNMP ifIndex 510) (Generation 149)
    Flags: Hardware-Down Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
    Protocol inet, MTU: 4470, Generation: 173, Route table: 0
      Flags: Sendbcast-pkt-to-re, Protocol-Down
      Input Filters: ipv4-filter
      Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
        Destination: 10.39/16, Local: 10.39.1.1, Broadcast: 10.39.255.255, Generation: 163
```

Displaying the Number of Packets Processed by the Policer at the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show firewall` operational mode command for the filter you applied to the logical interface.

```
user@host> show firewall filter ipv4-filter
Filter: ipv4-filter
Policers:
Name                                                Packets
```

```
shared-policer-A-tcp-police-1                    32863
shared-policer-A-tcp-police-2                     3870
```

The command output displays the name of policer (`shared-policer-A`), the name of the filter term (`police-1`) under which the policer action is specified, and the number of packets that matched the filter term. This is only the number of out-of-specification (out-of-spec) packet counts, not all packets policed by the policer.

RELATED DOCUMENTATION

- Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**
- Firewall Filter Match Conditions Based on Bit-Field Values | **976**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**
- Two-Color Policer Configuration Overview | **2034**
- Physical Interface Policer Overview

Firewall and Policing Differences Between PTX Series Packet Transport Routers and T Series Matrix Routers

This topic provides a list of firewall and policier features available on PTX Packet Transport Routers and compares them with firewall and policing features on T Series routers.

Firewall Filters

Junos OS firewall and policing software on PTX Series Packet Transport Routers supports IPv4 filters, IPv6 filters, MPLS filters, CCC filters, interface policing, LSP policing, MAC filtering, ARP policing, L2 policing, and other features. Exceptions are noted below.

- PTX Series Packet Transport Routers do not support:
  - Egress Forwarding Table Filters
  - Forwarding Table Filters for MPLS/CCC
  - Family VPLS
- PTX Series Packet Transport Routers do not support nested firewall filters. The `filter` statement at the [`edit firewall family` *family-name* `filter` *filter-name* `term` *term-name*] hierarchy level is disabled.

- Because no service PICs are present in PTX Series Packet Transport Routers, service filters are not supported for both IPv4 and IPv6 traffic. The `service-filter` statement at `[edit firewall family (inet | inet6)]` hierarchy level is disabled.
- The PTX Series Packet Transport Routers exclude simple filters. These filters are supported on Gigabit Ethernet intelligent queuing (IQ2) and Enhanced Queuing Dense Port Concentrator (EQ DPC) interfaces only. The `simple-filter` statement at the `[edit firewall family inet)]` hierarchy level is disabled.
- Physical interface filtering is not supported. The `physical-interface-filter` statement at the `[edit firewall family` *`family-name`* `filter` *`filter-name`*`]` hierarchy level is disabled.
- The prefix action feature is not supported on PTX Series Packet Transport Routers. The `prefix-action` statement at `[edit firewall family inet]` hierarchy level is disabled.
- On T Series routers, you can collect a variety of information about traffic passing through the device by setting up one or more accounting profiles that specify some common characteristics of the data. The PTX Series Packet Transport Routers do not support accounting configurations for firewall filters. The `accounting-profile` statement at the `[edit firewall family` *`family-name`* `filter` *`filter-name`*`]` hierarchy level is disabled.
- The `reject` action is not supported on the loopback (`lo0`) interface. If you apply a filter to the `lo0` interface and the filter includes a `reject` action, an error message appears.
- PTX Series Packet Transport Routers do not support aggregated ethernet logical interface match conditions. However, child link interface matching is supported.
- PTX Series Packet Transport Routers displays both counts if two different terms in a filter have the same match condition but they have different counts. T Series routers display one count only.
- PTX Series Packet Transport Routers do not have separate policer instances when a filter is bound to multiple interfaces. Use the `interface-specific` configuration statement to create the configuration.
- On PTX Series Packet Transport Routers, when an ingress interface has CCC encapsulation, packets coming in through the ingress CCC interface will not be processed by the egress filters.
- For CCC encapsulation, the PTX Series Packet Transport Routers append an extra 8 bytes for egress Layer 2 filtering. The T Series routers do not. Therefore, egress counters on PTX Series Packet Transport Routers show an extra eight bytes for each packet which impacts policer accuracy.
- On PTX Series Packet Transport Routers, output for the `show pfe statistics traffic` CLI command includes the packets discarded by DMAC and SMAC filtering. On T Series routers, the command output does not include these discarded packets because MAC filters are implemented in the PIC and not in the FPC.

- The last-fragment packet that goes through a PTX firewall cannot be matched by the `is-fragment` matching condition. This feature is supported on T Series routers.

  A possible workaround on PTX Series Packet Transport Routers is to configure two separate terms with same the actions: one term contains a match to `is-fragment` and the other term contains a match to `fragment-offset -except 0`.

- On PTX Series Packet Transport Routers, MAC pause frames are generated when packet discards exceed 100 Mbps. This occurs only for frame sizes that are less than 105 bytes.

Traffic Policiers

Junos OS firewall and policing software on PTX Series Packet Transport Routers supports IPv4 filters, IPv6 filters, MPLS filters, CCC filters, interface policing, LSP policing, MAC filtering, ARP policing, L2 policing, and other features. Exceptions are noted below.

- PTX Series Packet Transport Routers support ARP policing. T Series routers do not.
- PTX Series Packet Transport Routers do not support LSP policing.
- PTX Series Packet Transport Routers do not support the `hierarchical-policer` configuration statement. .
- PTX Series Packet Transport Routers do not support the `interface-set` configuration statement. This statement groups a number of interfaces into a single, named interface set.
- PTX Series Packet Transport Routers do not support the following policer types for both normal policers and three-color policers:
  - `logical-bandwidth-policer` — Policer uses logical interface bandwidth.
  - `physical-interface-policer` — Policer is a physical interface policer.
  - `shared-bandwidth-policer` — Share policer bandwidth among bundle links.
- When a policer action and forwarding-class, loss-priority actions are configured within the same rule (a *Multifield Classification*), the PTX Series Packet Transport Routers work differently than T Series routers. As shown below, you can configure two rules in the filter to make the PTX filter behave the same as the T Series filter:

  PTX Series configuration:

```
rule-1 {
    match: {x, y, z}
    action: {forwarding-class, loss-prio, next}
}
    rule-2 {
        match: {x, y, z}
```

```
        action: {policer}
    }
```

T Series configuration:

```
rule-1 {
    match: {x, y, z}
    action: {forwarding-class, loss-prio, policer}
}
```

RELATED DOCUMENTATION

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Hierarchical Policers on ACX Series Routers Overview

On ACX Series routers, two-level ingress hierarchical policing is supported. Single-level policers define a single bandwidth profile that is used by multiple traffic flows with different priorities. Two-level policers enable a single bandwidth profile to be optimally used for multiple traffic flows, based on bandwidth and priority needs of a network. Typically, multiple traffic flows can share a single policer instance. With single-level policers, you cannot adminster the method using which the committed information rate (CIR) and the peak information rate (PIR) values specified in the bandwidth profile are shared across different flows. For example, in a certain network deployment, you might want an equal or even distribution of CIR across the individual flows. In such a scenario, you cannot accomplish this requirement using single-level policers and need to configure aggregate or hierarchical policers.

NOTE: Hierarchical policers is not applicable on ACX5048 and ACX5096 routers.

Hierarchical policers control the sharing of an aggregate traffic rate across multiple micro-flows, which constitute the aggregate flow or the macro-flow. Micro-flows are defined and matched using firewall filter rules and the action of these rules point to a macro-policer. This macro- policer or aggregate policer determines the amount of aggregate bandwidth that can be used by the micro-flows that are associated with it. You can control the bandwidth to be utilized among the micro-flows in different ways.

NOTE: The hierarchical policing mechanism on ACX routers is different from the hierarching policing capability supported on MX Series routers. On MX Series routers, with a hierarchical policer, only one child or subordinate policer can be configured under a parent, top-level policer, whereas on ACX Series routers, you can aggregate and specify multiple child policers under a single parent policer. The hierarchical policing methodology on ACX routers is also called aggregate policing.

Policers are used to enforce bandwidth profiles on the transmitted traffic. A bandwidth profile is configured for each user based on the service level agreement (SLA) and the subscription plan that has been requested by the user from the service or enterprise provider. A bandwidth profile is defined using the following parameters:

- Committed information rate (CIR) denoted in bits per second (bps).
- Committed burst size (CBS) denoted in bytes.
- Excess information rate (EIR) denoted in bps.
- Excess burst size (EBS) denoted in bytes.
- Color mode (CM) can contain only one of two possible values, color-blind or color-aware. In color-aware mode, the local router can assign a higher packet loss priority, but cannot assign a lower packet loss priority. In color-blind mode, the local router ignores the preclassification of packets and can assign a higher or lower packet loss priority.

A policer is then used to enforce the bandwidth profile and perform different actions, depending on whether a certain packet confirms to the attributes in the bandwidth profile or does not satisfy the values in the configured bandwidth profile. Hierarchical policers can be considered to be an alternative technique for hierarchical queuing and shaping. However, a few differences exist between the operations that a hierarchical policer performs when matched against the processes that a hierarchical scheduler performs.

Hierarchical scheduler enables fine-grained bandwidth sharing in terms of percentages of the available bandwidth, whereas hierarchical policing only enables a coarse-grained bandwidth sharing based on the absolute micro-flow values of CIR and EIR. Hierarchical policing enables the packet loss priority (PLP) and also the forwarding class to be modified in certain cases, depending on whether the packet is confirming, exceeding, or violating the particular bandwidth profile. Hierarchical scheduler does not cause any modifications to the PLP or forwarding class values of a packet. Modifications are performed only for violating packets.

ACX routers do not support hierarchical queuing and shaping. Ingress hierarchical policers can work in conjunction with ingress, egress, or both ingress and egress hierarchical queues. For example, a two-

level ingress hierarchical policer combined with a two-level egress queuing framework results in a four-level CoS capability.

RELATED DOCUMENTATION

Guidelines for Configuring Hierarchical Policers on ACX Series Routers | 1951
Hierarchical Policer Modes on ACX Series Routers | 1953
Processing of Hierarchical Policers on ACX Series Routers | 1958
Actions Performed for Hierarchical Policers on ACX Series Routers | 1959
Configuring Aggregate Parent and Child Policers on ACX Series Routers | 1962

Guidelines for Configuring Hierarchical Policers on ACX Series Routers

Keep the following points in mind when you configure hierarchical or aggregate policers:

- You cannot specify the same policer as both a child policer and a parent policer.
- The child policers of a hierarchical policer use the same resources as normal policers. Therefore, the maximum number of child policers and normal policers in the system for bridge domains and IPv4 services is as follows:
  - Family Bridge

    A group of 124 entries of policers shared with other family-bridge filters.

    A maximum of approximately 62 policers when no other family-bridge filters with the count action for the firewall filter.

    Along with 62 policers, you can configure up to 62 family-bridge filters without the count action for the firewall filter.
  - Family inet

    A group of 250 entries of policers shared with other family-inet filters.

    A maximum of about 125 policers when no other family-inet filters with the count action.

    Along with 125 policers, you can define up to 62 family-inet filters without the count action.
- The hierarchical policer supports the same policer ranger and burst size behavior as normal policers.
- You must configure the same hierarchical policer mode for all child policers that refer or link with the same parent policer instance.

- You cannot use the same policer template as both a normal policer and a child policer.
- You cannot use the same base policer settings as both a normal policer and an aggregate or hierarchical policer.
- You cannot use the filter-specific statement at the `[edit firewall policer policer-name]` hierarchy level to instantiate an aggregate policer. Instead, the instantiation of the policer is performed by including the aggregate statement at the `[edit firewall policer policer-name]` hierarchy level.
- All the child policers of a certain aggregate policer must contain the same definitions of settings or attributes.
- All the policer instantiation formats are supported for the aggregate policer.
- Ingress two-level hierarchical policing is supported on all ACX routers.
- All the supported match conditions for filters used with normal policers are supported with micro-flow policers.
- You can configure up to 32 hierarchical policer instances. You can configure up to 32 child policers per macro policer instance.
- You can configure hierarchical policers on aggregated Ethernet interfaces.

NOTE: Hierarchical policers is not applicable on ACX5048 and ACX5096 routers.

RELATED DOCUMENTATION

Hierarchical Policers on ACX Series Routers Overview | **1949**

Hierarchical Policer Modes on ACX Series Routers | **1953**

Processing of Hierarchical Policers on ACX Series Routers | **1958**

Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**

Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Hierarchical Policer Modes on ACX Series Routers

IN THIS SECTION

- Guarantee Mode | 1953
- Peak Mode | 1955
- Hybrid Mode | 1956

The method in which the micro-flow policer determines and manages the share of the aggregate bandwidth for the micro-flow is defined by the hierarchical policer mode. ACX routers support the following three hierarchical policer modes. You can configure the mode or type of the policer for each hierarchical policer instance.

NOTE: Hierarchical policer is not applicable on ACX5048 and ACX5096 routers.

Guarantee Mode

This mode, also called bandwidth-guarantee mode, is used when the micro-flow policer is used to specify that a portion of the aggregate parent policer bandwidth is guaranteed for its micro-flow. When this micro-flow contains no traffic, then amount allocated for this micro-flow out of the aggregate bandwidth is used by the other micro-flows that are transmitting traffic with a size-limit or bandwidth that is higher than their respective guaranteed bandwidth rates.

Consider a sample scenario in which the maximum allowed rate or peak information rate (PIR) for a user is 140 Mbps. A total of four services or applications called expedited forwarding (EF), Gold, Silver and Bronze are defined for the guaranteed bandwidth mode of policer with a CIR of 50 Mbps, 40 Mbps, 30 Mbps, and 20 Mbps respectively. For example, if 140 Mbps of trafic is received for each of the four services, then the permitted traffic rates are 50, 40, 30 and 20 Mbps respectively. If 150 Mbps of Gold traffic is received, only 140 Mbps is permitted for Gold traffic.

All the child policers must be of single-rate, single-bucket, and two-color modes for bandwidth guarantee mode of hiearchical policer. This combination of attributes is also called floor mode. The micro-flow policer value specifies the minimum guaranteed bandwidth (CIR) for the micro-flow. The macro-flow policer value specifies the maximum allowed bandwidth (PIR) for all the flows. The sum or the cumulative value of all CIR values of the configured micro-flows must be less than or equal to the macro-flow PIR. The burst size of macro-flow must be greater than the sum of the aggregate of the

burst size of all the child policers and the largest MTU of the physical interface among all the physical interfaces of the logical interfaces or interface families to which the child policers are attached.

Consider a sample configuration that has two child policers aggregated by a parent PIR in bandwidth-guarantee mode. PIRs for the children policers and the parent policer are configured. When two flows, flow 1 and flow 2, transmit traffic at a rate that exceeds the configured PIR values, then the share of the parent PIR is adjusted to permit traffic for the child policers based on their priorities defined for the flows, while the bandwidth is maintained.

Policers use a token bucket algorithm to enforce a limit on an average transmit or receive rate of traffic at an interface while allowing bursts of traffic up to a maximum value based on the configured bandwidth limit and configured burst size. The token bucket algorithm offers more flexibility than a leaky bucket algorithm in that you can allow a specified traffic burst before starting to discard packets or apply a penalty such as packet output-queuing priority or packet-drop priority. Following are the main components of the token bucket algorithm:

- The bucket represents a rate-limiting function of the policer on the interface input or output traffic.
- Each token in the bucket represents a “credit” for some number of bits, and tokens in the bucket are “cashed in” for the ability to receive or transmit traffic that conforms to a rate limit configured for the policer.
- The token arrival rate is a periodic allocation of tokens into the token bucket that is calculated from the configured bandwidth limit.
- The token bucket depth defines the capacity of the bucket in bytes. Tokens that are allocated after the bucket reaches capacity are not able to be stored and used.

An arriving packet complies with the bandwidth-guarantee mode if tokens are present in the peak burst size (PBS) of either the parent policer or the committed burst size (CBS) of the child policer. If sufficient tokens are not present in the PBS or CBS of either of the parent or child policers respectively, the packet does not conform to the guarantee mode of the hierarchical policer working. In such a case, the child policer rate is guaranteed for the member flows. The following table describes the different scenarios of color-coding for micro-flow and macro-flow policers and the resultant color or priority that is assigned:

Micro-Color	Macro-Color	Result
Green	Green	Green
Green	Red	Green
Red	Green	Green

(Continued)

Micro-Color	Macro-Color	Result
Red	Red	Red

Peak Mode

This mode, also called bandwidth-protection mode, is used when the micro-flow policer is used to specify the maximum amount of the aggregate parent policer bandwidth that the micro-flow can use. This mode is used to protect a given micro-flow from starving the other flows. Even when the other micro-flows contain no traffic (the available aggregate bandwidth rate is greater than the rate of the particular micro-flow, the micro-flow cannot use more than the rate configured on its micro-flow policer.

Consider a sample scenario in which the total maximum allowed rate (PIR) for a user is 100 Mbps. A total of four services or applications called expedited forwarding (EF), Gold, Silver and Bronze are defined for the peak or bandwidth-restriction mode of the policer with PIR values of 50 Mbps, 40 Mbps, 30 Mbps, and 20 Mbps respectively. Such a setting is used in topologies in which you want to prevent a certain subscriber or user from utilizing an increased share ofthe macro-flow or the parent CIR for real-time applications, such as video-on-demand (VoD) or voice over IP (VoIP). For example, if only 100 Mbps of EF packets are received, the permitted bandwidth rate for the traffic is 50 Mbps. When 100 Mbps of traffic is received for each of the four services, then the aggregate allowed traffic is 100 Mbps, in which the rates are as follows for the different services:

- Less than or equal to 50 Mbps for EF traffic
- Less than or equal to 40 Mbps for Gold traffic
- Less than or equal to 30 Mbps for Silver traffic
- Less than or equal to 20 Mbps for Bronze traffic

All the child policers must be of single-rate, single-bucket, and two-color types for bandwidth-protection or peak mode of the hierarchical policer. The micro-flow policer value specifies the maximum allowed bandwidth (PIR) for the micro-flow. The macro-flow policer value specifies the maximum allowed bandwidth (PIR) for all the flows. The sum of the PIR values of the micro-flows must be greater than or equal to the PIR values of the child policers. The macro-flow burst-size must be greater than or equal to that of the micro-flow with the largest burst-size.

Consider a sample configuration that has two child policers aggregated by a parent PIR in bandwidth-guarantee mode. PIRs for the children policers and the parent policer are configured. When two flows, flow 1 and flow 2, transmit traffic at a rate that exceeds the configured PIR values, then the share of the

parent PIR is adjusted to permit traffic for the child policers based on their priorities defined for the flows, while the bandwidth is restricted to maintain the minimum or committed rates of traffic flows.

An arriving packet complies with the bandwidth-guarantee mode if tokens are present in the peak burst size (PBS) of both the child policer and the parent policer. If sufficient tokens are not present in the PBS of both the policers, the packet does not conform to the peak mode of the hierarchical policer working. In such a case, the child policer rate is the maximum allowed rate or PIR for the member flows. The following table describes the different scenarios of color-coding for micro-flow and macro-flow policers and the resultant color or priority that is assigned:

Micro-Color	Macro-Color	Result
Green	Green	Green
Green	Red	Red
Red	Green	Red
Red	Red	Red

Hybrid Mode

This mode, which is a combination of the bandwidth-guarantee and bandwidth-protection modes, enables the capabilities of bandwidth restriction and the per-flow bandwidth moderation to be accomplished simultanouesly. Bandwidth-guarentee or bandwidth-restriction mode controls the guaranteed rates for a given micro-flow. However, it does not adminster or manage the manner in which the excess aggregate bandwidth can be shared among the micro-flows. A certain micro-flow can potentially use all the excess aggregate bandwidth starving the other micro-flows of any excess bandwidth.

Bandwidth-protection or peak mode controls the amount of bandwidth that a particular micro-flow can consume, thereby protecting other flows from being starved. However, it does not specify any guaranteed rates for the micro-flows. For example, if micro-flow rates for flows f1, f2 and f3 are 50 Mbps, 60 Mbps, 50 Mbps respectively, and the aggregate rate is 70 Mbps, it is possible that f1 and f2 flows might be provided 50 Mbps and 20 Mbps respectively, with no bandwidth allocated for f3.

Hybrid mode implements the benefits of the peak and guaranteed modes to overcome their individual limitations. In hybird mode, the micro-flow policer specifies two rates, CIR and EIR, for the micro-flow. The CIR specifies the guaranteed portion out of the total macro-flow bandwidth for a micro-flow, and the PIR specifies the maximum portion of the total macro-flow bandwidth for a micro-flow. This

mechanism is analogues to CIR functioning in guarantee mode and EIR functioning in peak mode, thereby combining the advantages of both models. In hyrbid mode, both color-aware and color-blind modes are supported for child policers.

Child policers operate in compliance with the RFC 4115 mode of two-rate three color markers. Normal two-rate three color markers on ACX routers operate in compliance with the RFC2698 mode.

Consider a sample configuration in which the maximum allowed rate for a user is 140 Mbps. A total of four services or applications called expedited forwarding (EF), Gold, Silver and Bronze are defined for the hybrid mode of the policer with PIR values of 55Mbps, 60 Mbps, 130 Mbps, and 140 Mbps respectively. The defined CIR values are 50 Mbps, 40 Mbps, 30 Mbps, and 20 Mbps for EF, Gold, Silver, and Bronze services respectively. For example, when 140 Mbps of traffic is received for each of the four services, then the permitted green-colored traffic is 50, 40, 30 and 20 Mbps respectively for the four services. If only 140 Mbps of EF traffic is received, 50 Mbps of EF traffic as green and 5 Mbps of EF traffic as yellow are permitted. In the same scenario, assume the macro-policer rate to be 26 Mbps. Also, assume two child policers in color-aware mode, namely, child policer-1 with a CIR of 10 Mbps and an EIR of 10 Mbps. Child policer-2 has a CIR of 15 Mbps and an EIR of 5 Mbps. When flow-1 is a 100 Mbps stream of yellow traffic, and flow-2 is an 100 Mbps stream of green traffic, the output of this policer hierarchy is as follows:

- Flow-1 has 0 Mbps of green traffic and has less than or equal to 5 Mbps of yellow traffic.
- Flow-2 has 10 Mbps of green traffic and has greater than or equal to 10 Mbps of yellow traffic.
- The sum of yellow traffic is less than or equal to 11 Mbps .

Consider a sample configuration that has two child policers aggregated by a parent PIR in hybrid mode. PIRs for the children policers and the parent policer are configured. When two flows, flow 1 and flow 2, transmit traffic at a rate that exceeds the configured PIR values, then the share of the parent PIR is adjusted to permit traffic for the child policers while the child PIR values are preserved for the two flows.

Hybrid mode of working of the aggregate or hierarchical policer supports two rates (CIR and PIR) and three colors for micro-flows. On ACX routers, for hybrid type of the policer, the micro-policer must be of type modified-trtcm as defined in RFC 4115. Both color-blind and color- aware modes are supported for child policers. Macro policer must be a single rate, single bucket, two color policer with the sum of the CIR values of the micro-flows being less than the PIR value of the macro-flow, and the cumulative value of all the PIR values of the micro-flows being greater than the PIR value of the macro-flow. When micro-flow traffic is less than the CIR value of the micro-flow CIR, the policer causes either the micro-flow CIR to be maintained or PIR to be achieved. When micro-flow traffic is greater than the CIR value of the micro-flow, the micro-flow CIR is guaranteed. Micro-flow excess rates are shared based on the available macro-flow bandwidth with the limitation of the excess information rate distributed for the micro-flows being implemented by the micro-flow PIR. The CBS of the macro-flow must be greater than or equal to the aggregate of the micro-flow CBS. The excess burst size (EBS) of the macro-flow must be greater than or equal to that of the micro-flow with the largest EBS.

An arriving packet complies with the hybrid mode if tokens are present in the committed burst size (CBS) of the child policer. The packet does not comply with hybrid mode if tokens are present in both the EBS of the child policer and the PBS of the parent policer. When a packet does not satisfy the hybrid mode of working of a policer, the CIR of the child policer is guaranteed for the member traffic flows and the PIR value of the child policer is the maximum permitted rate for the member flows. The following table describes the different scenarios of color-coding for micro-flow and macro-flow policers and the resultant color or priority that is assigned:

Micro-Color	Macro-Color	Result
Green	Green	Green
Red	Green	Green
Yellow	Green	Yellow
Yellow	Red	Red
Red	Green	Red
Red	Red	Red

RELATED DOCUMENTATION

- Hierarchical Policers on ACX Series Routers Overview | **1949**
- Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**
- Processing of Hierarchical Policers on ACX Series Routers | **1958**
- Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**
- Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Processing of Hierarchical Policers on ACX Series Routers

Hierarchical policer provisions a controlled sharing of an aggregate among micro-flows. For example, a hierarchical policer is used to share the bandwidth of a certain subscriber or user across different CoS

settings of that user. Assume that the user is configured to connect using a logical interface, and the CoS functionality is configured using DiffServ code point (DSCP) in the traversed packet. The user is assigned an aggregate bandwidth of 140 Mbps. This consolidated bandwidth must be distributed and shared amongst the four supported CoS settings represented by DSCP values of 11, 12, 21, and 22 respectively in the order of 50 Mbps, 40 Mbps, 30 Mbps and 20 Mbps. To obtain this behavior, you must perform the following configurations:

- Configure micro-flows--A micro-flow is characterized by all packets that pass through the same micro policer (or child policer) instance. To enable this configuration, packets must be classified as micro-flows by using filters to match the packets, and the action of the filter to refer to the macro policer. You can group and combine packets matching multiple different filters or terms into a single micro-flow by specifying the same policer instance across the different filters and terms. These settings are identical to the configuration required to associate a single-level policer with a filter.

- Assign child policers to the aggregate policer--This step is additional, from the procedure to create a single-level policer, to configure a hierarchical policer. You must link or associate the child policers to the parent or aggregate policer. You can perform this linking by specifying at each child policer by using the `aggregate-policing` *`aggregate-policer-name`* statement at the `[edit firewall policer` *`policer-name`*`]` hierarchy level.

NOTE: Hierarchical policer is not applicable on ACX5048 and ACX5096 routers.

RELATED DOCUMENTATION

Hierarchical Policers on ACX Series Routers Overview | **1949**

Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**

Hierarchical Policer Modes on ACX Series Routers | **1953**

Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**

Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Actions Performed for Hierarchical Policers on ACX Series Routers

IN THIS SECTION

Instantiation Methods for Child and Aggregate Policers | **1960**

Instantiation Methods for Child Policers or Normal Policers | 1960
Instantiation Methods for Aggregate Policers | 1961

The hierarchical parent policer impacts the packet loss priority (PLP) of the child policer. The PLP-based actions defined in the then statement of the are performed, based on the PLP derived from the combined processing of the child and parent policers. The then statement defined in the parent policer is not effective. This section contains the following topics that describe the methods of instantiation of aggregate or hierarchical policers and child or normal policers.

NOTE: Hierarchical policer is not applicable on ACX5048 and ACX5096 routers.

Instantiation Methods for Child and Aggregate Policers

In the Junos OS, a certain policer configuration or a policer template is used to create multiple instances of the policer in the hardware using the template attributes such as the CIR, PIR, CBS, and PBS values specified in the template. You need not create multiple policer configurations with the same attributes for effective management by using aggregate policers.

Instantiation Methods for Child Policers or Normal Policers

For a normal policer or a child policer, if you specify a filter-specific attribute for a policer by entering the `filter-specific` statement at the `[edit firewall policer policer-name]` or `[edit logical-systems logical-system-name firewall policer policer-name]` hierarchy level, when you specify a 'filter-specific' clause, a single policer instance is used by all terms (within the same firewall filter) that reference the policer. For example, if a filter F1 contains terms T1 and T2, they refer to the same policer, say P1. If you configure the policer P1 as a filter-specific type, the same policer instance on the device is referred by both the terms T1 and T2. In this case, F1 is attached to a logical interface named IFL1, which is configured on the physical interface named IFD1.

By default, a policer operates in term-specific mode so that, for a given firewall filter, the Junos OS creates a separate policer instance for every filter term that references the policer. This operation is the default instantiation mode if you do not configure the `filter-specific` statement. For example, consider a filter F1 that has two terms, T1 and T2. Both these terms refer to the same policer, namely P1. With a term-specific mode of the policer, two policer instances are created on the device, one each for terms T1 and T2.

Instantiation Methods for Aggregate Policers

The following modes of operations are possible for aggregate policers.

- Global—Shares the same parent policer across all the child policer instances in the system.
- Physical interface-specific—Shares the same parent policer across all the child policer instances of a certain physical interface. This mode is not supported on ACX routers.
- Filter-specific—Shares the same parent policer across all the child policer instances of the same filter. This mode is not supported on ACX routers.
- Interface-specific—Shares the same parent policer across all the child policer instances of the same logical interface. This mode is not supported on ACX routers.

You can include the `aggregate global` statement at the `[edit firewall policer policer-template-name]` hierarchy level to define an aggregate policer to be shared or applicable across all of the child policer instances in the router. You can include the aggregate parent statement at the [edit firewall policer policer-template-name] hierarchy level to define an aggregate policer as the parent policer. The following statement does not take effect for aggregate policers: `set firewall policer` *`policer-template-name`* `filter-specific`.

Consider a sample deployment in which an aggregate policer named P3 is configured. P1 and P2 are child policers. T1, T2, T3, and T4 are terms. F1 and F2 are filters. Logical interfaces, IFL1 and IFL2, are created on the underlying physical interfaces, IFD1 and IFD2 in this configuration. Interface address families are correspondingly created on the interfaces as IFF1 and IFF2.

If you configure an interface-specific filter, term-specific child policer, and the aggregate policer as the global policer, a single instance of P3 is created and associated with the child policers, P1 and P2. Two instances each of P1 and P2 are created, one for T1 within F1 and the other for T2 within F1. The two instances each of P1 and P2 are associated with IFL1 and IFL2, which in turn are bound to IFD1 and IFD2.

If you configure an interface-specific filter, term-specific or filter-specific child policer, and the aggregate policer is physical interface- specific policer, two instances of P3 are created. One instance of P3 contains two child policer instances of P1. P1 contains the filter F1 and term T1 to be applied to IFL1 and IFL2. The other instance of P3 contains two child policer instances of P1. P1 contains F1 and T1 to be applied to another two logical interfaces, IFL3 and IFL4.

If you configure an interface-specific filter, term-specific child policer, and interface-specific aggregate policer, two instances of P3 are created. Each P3 instance contains two P1 instances. One P1 instance contains F1 and T1 for IFF1 to be applied to IFL1. The other P1 instance contains F2 for IFF2 to be applied to IFL1. The other P3 instance contains two P1 instances. Here, one P1 contains F1 and T1 for IFF3 and the other P1 contains F2 and T1 for IFF4 to be applied on IFL2.

If you configure an interface-specific filter, term-specific child policer, and filter-specific aggregate policer, two P3 instances are created. Each P3 contains two P1 instances. One P1 contains P1, T1, F1

IFF1, applied to IFL1. The other P1 contains P1, T2, F1, IFF1, applied to IFL1. The third P1 contains P1, T3, F2, IFF2, applied to IFL1. The last P1 contains P1, T4, F2, IFF2, applied to IFL1.

RELATED DOCUMENTATION

- Hierarchical Policers on ACX Series Routers Overview | **1949**
- Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**
- Hierarchical Policer Modes on ACX Series Routers | **1953**
- Processing of Hierarchical Policers on ACX Series Routers | **1958**
- Configuring Aggregate Parent and Child Policers on ACX Series Routers | **1962**

Configuring Aggregate Parent and Child Policers on ACX Series Routers

On ACX Series routers, two-level ingress hierarchical policing is supported. Single-level policers define a single bandwidth profile. You must first define the child or subordinate policers and associate or link them with the aggregate parent policer, which is globally applicable for the entire system. You can configure the mode of hierarchical or aggregate policing for the child policers, such as peak mode, guarantee mode, or hybrid mode of policing.

NOTE: Hierarchical policer is not applicable on ACX5048 and ACX5096 routers.

NOTE: The hierarchical policing mechanism on ACX routers is different from the hierarching policing capability supported on MX Series routers. On MX Series routers, with a hierarchical policer, only one child or subordinate policer can be configured under a parent, top-level policer, whereas on ACX Series routers, you can aggregate and specify multiple child policers under a single parent policer under the `[edit firewall]` hierarchy level. The hierarchical policing methodology on ACX routers is also called aggregate policing. The `hierarchical-policer` statement and its substatements at the `[edit firewall]` hierarchy level that are supported on MX Series routers are not available for ACX Series routers.

To configure child or micro policers for an aggregate parent policer and associate the parent policer with the child policers:

1. Configure one normal policer as a child policer and specify the aggregate policing mode.

```
user@host# set policer mi_pol_1 if-exceeding bandwidth-limit 25m
user@host# set policer mi_pol_1 if-exceeding burst-size-limit 3k
user@host# set policer mi_pol_1 if-exceeding aggregate-policing policer mi_pol_x aggregate-
sharing-mode peak;
user@host# set policer mi_pol_1 then discard
```

2. Configure another normal policer as a child policer and specify the aggregate policing mode. The `aggregate-sharing-mode` option is a Packet Forwarding Engine statement.

```
user@host# set policer mi_pol_2 if-exceeding bandwidth-limit 30m
user@host# set policer mi_pol_2 if-exceeding burst-size-limit 30k
user@host# set policer mi_pol_2 if-exceeding aggregate-policing policer mi_pol_x aggregate-
sharing-mode peak;
user@host# set policer mi_pol_2 then discard
```

3. Define the aggregate parent policer as the global policer for the system. The `aggregate-sharing-mode` option is a Packet Forwarding Engine statement.

```
user@host# set policer mi_pol_x if-exceeding bandwidth-limit 55m
user@host# set policer mi_pol_x if-exceeding burst-size-limit 35k
user@host# set policer mi_pol_x aggregate global
```

4. Verify the settings of all policer templates configured by using the `show filter policer template` command.

```
user@host# show filter policer template
AppType Template name       Bw limit-bits/sec      Burst-bytes     Action Options
------- -------------       -----------------      -----------     --------------
      0 mi_pol_1
                            25000000                3000             DROP
         Aggregate Child of mi_pol_x mode=2
      0 mi_pol_2
                            30000000                30000            DROP
         Aggregate Child of mi_pol_x mode=2
      0 mi_pol_x
                            55000000                35000            DROP
         Aggregate Parent
```

5. View the configured policer instances that are linked to the aggregate parent policer by using the `show filter aggregate-policer` command.

```
user@host# show filter aggregate-policer p1
CHILDREN
-------
#1) [UNI1_filtermi_pol_trtcm1-t2]  CBS[1000]kB; CIR[10000]kbps;  CBS[2000]kB; PIR[30000]kbps;
Agg mode = 3;
#2) [UNI2_filtermi_pol_trtcm2-t2]  CBS[1000]kB; CIR[15000]kbps;  CBS[2000]kB; PIR[35000]kbps;
Agg mode = 3;


PARENT
------
[p1]  PBS[3000]kB; PIR[38000]kbps;
Sum child CIR[25000]kbps;CBS[2000]kB;
Sum child PIR[65000]kbps;PBS[4000]kB;
Max child CIR[15000]kbps;CBS[1000]kB;
Max child PIR[35000]kbps;PBS[2000]kB;


RESULTS
-------


STATUS = OK
```

The `show filter policer template` and `show filter aggregate-policer` CLI commands need to be run at the PFE level. To go to the PFE level, you need to:

1. Enter the `start shell` CLI command.

```
user@host> start shell
```

2. Establish a vty session by entering the `vty` shell command followed by the executable name for the component. For example, `vty feb0`.

```
user@host% vty feb0
```

3. Type the `show filter policer ...` CLI command.

RELATED DOCUMENTATION

- Hierarchical Policers on ACX Series Routers Overview | **1949**
- Guidelines for Configuring Hierarchical Policers on ACX Series Routers | **1951**
- Hierarchical Policer Modes on ACX Series Routers | **1953**
- Processing of Hierarchical Policers on ACX Series Routers | **1958**
- Actions Performed for Hierarchical Policers on ACX Series Routers | **1959**

CHAPTER 30

Configuring Policer Rate Limits and Actions

IN THIS CHAPTER

- Policer Bandwidth and Burst-Size Limits | 1966
- Policer Color-Marking and Actions | 1968
- Single Token Bucket Algorithm | 1971
- Dual Token Bucket Algorithms | 1974

Policer Bandwidth and Burst-Size Limits

Table 113 on page 1966 lists each of the Junos OS policer types supported. For each policer type, the table summarizes the bandwidth limits and burst-size limits used to rate-limit traffic.

Table 113: Policer Bandwidth Limits and Burst-Size Limits

Policer Type	Bandwidth Limits	Burst-Size Limits
Single-Rate Two-Color Policer		
Defines a single rate limit: a bandwidth limit and an allowed burst size for conforming traffic. For a single-rate two-color policer only, you can specify the bandwidth limit as a percentage value from 1 through 100 instead of as an absolute number of bits per second. The effective bandwidth limit is calculated as a percentage of either the physical interface media rate or the logical interface configured shaping rate.	`bandwidth-limit` *bps*; M and T Series routers: 8000..100000000000 MX Series routers: 8000..18446744073709551615	`burst-size-limit` *bytes*; M, MX, and T Series routers: 1500..10000000000
	`bandwidth-percent` 1..100 percent	

Table 113: Policer Bandwidth Limits and Burst-Size Limits *(Continued)*

Policer Type	Bandwidth Limits	Burst-Size Limits
Single-Rate Three-Color Policer		
Defines a single rate limit: a bandwidth limit and an allowed burst size for conforming traffic. Also defines a second, larger burst size. This second burst size is used to differentiate between two categories of nonconforming traffic (yellow or red).	`committed-information-rate` *bps*; M and T Series routers: 1500..100000000000 MX Series routers: 8000..1844674407370955161 5	`committed-burst-size` *bytes*; M, MX, and T Series routers: 1500..100000000000
		`excess-burst-size` *bytes*; M, MX, and T Series routers: 1500..100000000000
Two-Rate Three-Color Policer		
Defines a committed rate limit: a bandwidth limit and an allowed burst size for conforming traffic. Also defines a peak rate limit: a second, larger burst size and a second, higher bandwidth limit. These additional rate-limit components are used to differentiate between two categories of nonconforming traffic (yellow or red).	`committed-information-rate` *bps*; M and T Series routers: 1500..100000000000 MX Series routers: 8000..1844674407370955161 5	`committed-burst-size` *bytes*; M, MX, and T Series routers: 1500..100000000000
	`peak-information-rate` *bps*; M and T Series routers: 1500..100000000000 MX Series routers: 8000..1844674407370955161 5	`peak-burst-size` *bytes*; M, MX, and T Series routers: 1500..100000000000
Hierarchical Policer		

Table 113: Policer Bandwidth Limits and Burst-Size Limits *(Continued)*

Policer Type	Bandwidth Limits	Burst-Size Limits
Defines two policers, each with a bandwidth limit and an allowed burst size for conforming traffic. Different policing actions are applied based on whether the packets are classified for expedited forwarding (EF) or for a lower priority. Rate-limits ingress Layer 2 traffic at a SONET physical or logical interface hosted on supported routing platforms only.	`bandwidth-limit` *bps*; M40e, M120, and M320 (with FFPC and SFPC) edge routers and T320, T640, and T1600 core routers with Enhanced Intelligent Queuing (IQE) PICs: 32000..50000000000 MX Series routers: 8000..1844674407370955161 5	`burst-size-limit` *bytes*; M40e, M120, and M320 (with FFPC and SFPC) edge routers and T320, T640, and T1600 core routers with Enhanced Intelligent Queuing (IQE) PICs: 1500..2147450880

RELATED DOCUMENTATION

Policer Color-Marking and Actions | **1968**

Determining Proper Burst Size for Traffic Policers | **1909**

Policer Color-Marking and Actions

Table 114 on page 1968 lists each of the Junos OS policer types supported. For each policer type, the table summarizes the color-marking criteria used to categorize a traffic flow and, for each color, the actions taken on packets in that type of traffic flow.

Table 114: Implicit and Configurable Policer Actions Based on Color Marking

Policer Rate Limits and Color Marking	Implicit Action	Configurable Actions
Single-Rate Two-Color Policer • Bandwidth limit • Burst size		

Table 114: Implicit and Configurable Policer Actions Based on Color Marking *(Continued)*

Policer Rate Limits and Color Marking		Implicit Action	Configurable Actions
	Green Conforms to rate and burst size limits	Set PLP to `low`	–
	Red Exceeds rate and burst size limits	–	• Discard the packet. • Assign to a forwarding class. • Set PLP to `low` or `high`. On some platforms, you can also set the PLP to `medium-low` or `medium-high`.
Single-Rate Three-Color Policer • Committed information rate (CIR) • Committed burst size (CBS) • Excess burst size (EBS)			
	Green Conforms to the CIR and CBS	Set PLP to `low`	–
	Yellow Exceeds the CIR and CBS but conforms to the EBS	Set PLP to `medium-high`	–
	Red Exceeds the EBS	Set PLP to `high`	• Discard the packet.

Table 114: Implicit and Configurable Policer Actions Based on Color Marking *(Continued)*

Policer Rate Limits and Color Marking		Implicit Action	Configurable Actions
Two-Rate Three-Color Policer • Committed information rate (CIR) • Committed burst size (CBS) • Peak information rate (PIR) • Peak burst size (PBS)			
	Green Conforms to the CIR and CBS	Set PLP to `low`	–
	Yellow Exceeds the CIR and CBS, but conforms to the PIR	Set PLP to `medium-high`	–
	Red Exceeds the PIR and PBS	Set PLP to `high`	• Discard the packet.
Hierarchical Policer			
Aggregate policer • Bandwidth limit • Burst size			
	Green Conforms to rate limits	Set PLP to `low`	–

Table 114: Implicit and Configurable Policer Actions Based on Color Marking *(Continued)*

Policer Rate Limits and Color Marking	Implicit Action	Configurable Actions
Red Exceeds rate limits	–	• Discard the packet. • Assign to a forwarding class. • Set PLP to `low` or `high`. On some platforms, you can also set the PLP to `medium-low` or `medium-high`.
Premium policer • Bandwidth limit • Burst size		
Green Conforms to rate limits	Set PLP to `low`	–
Red Exceeds rate limits	–	• Discard the packet.

RELATED DOCUMENTATION

Policer Bandwidth and Burst-Size Limits | **1966**

Determining Proper Burst Size for Traffic Policers | **1909**

Single Token Bucket Algorithm

IN THIS SECTION

Token Bucket Concepts | **1972**

Single Token Bucket Algorithm | 1972
Conformance Measurement for Two-Color Marking | 1973

Token Bucket Concepts

When you apply traffic policing to the input or output traffic at an interface, the rate limits and actions specified in the policer configuration are used to enforce a limit on the average throughput rate at the interface while also allowing bursts of traffic up to a maximum number of bytes based on the overall traffic load. Junos OS policers measure traffic-flow conformance to a policing rate limit by using a *token bucket algorithm*. An algorithm based on a single token bucket allows burst of traffic for short periods, whereas an algorithm based dual token buckets allows more sustained bursts of traffic.

Single Token Bucket Algorithm

A single-rate two-color policer limits traffic throughput at an interface based on how the traffic conforms to rate-limit values specified in the policer configuration. Similarly, a hierarchical policer limits traffic throughput at an interface based on how aggregate and premium traffic subflows conform to aggregate and premium rate-limit values specified in the policer configuration. For both two-color policer types, packets in a conforming traffic flow are categorized as *green*, and packets in a non-conforming traffic flow are categorized as *red*.

The single token bucket algorithm measures traffic-flow conformance to a two-color policer rate limit as follows:

- The token arrival rate represents the single *bandwidth limit* configured for the policer. You can specify the bandwidth limit as an absolute number of bits per second by including the `bandwidth-limit` *`bps`* statement. Alternatively, for single-rate two-color policers only, you can use the `bandwidth-percent` *`percentage`* statement to specify the bandwidth limit as a percentage of either the physical interface port speed or the configured *logical interface* shaping rate.
- The token bucket depth represents the single *burst size* configured for the policer. You specify the burst size by including the `burst-size-limit` *`bytes`* statement.
- If the bucket is filled to capacity, arriving tokens “overflow” the bucket and are lost.

When the bucket contains insufficient tokens for receiving or transmitting the traffic at the interface, packets might be dropped or else re-marked with a lower forwarding class, a higher packet loss priority (PLP) level, or both.

Conformance Measurement for Two-Color Marking

In two-color-marking policing, a traffic flow whose average arrival or departure rate does not exceed the token arrival rate (bandwidth limit) is considered *conforming traffic*. Packets in a conforming traffic flow (categorized as green traffic) are implicitly marked with a packet loss priority (PLP) level of `low` and then passed through the interface.

For a traffic flow whose average arrival or departure rate exceeds the token arrival rate, conformance to a two-color policer rate limit depends on the tokens in the bucket. If sufficient tokens remain in the bucket, the flow is considered conforming traffic. If the bucket does not contain sufficient tokens, the flow is considered *non-conforming traffic*. Packets in a non-conforming traffic flow (categorized as red traffic) are handled according to policing actions. Depending on the configuration of the two-color policer, packets might be implicitly discarded; or the packets might be re-marked with a specified forwarding class, a specified PLP, or both, and then passed through the interface.

NOTE: The number of tokens remaining in the bucket at any given time is a function of the token bucket depth and the overall traffic load.

The token bucket is initially filled to capacity, and so the policer allows an initial traffic burst (back-to-back traffic at average rates that exceed the token arrival rate) up to the size of the token bucket depth.

During periods of relatively low traffic (traffic that arrives at or departs from the interface at average rates below the token arrival rate), unused tokens accumulate in the bucket, but only up to the configured token bucket depth.

RELATED DOCUMENTATION

Two-Color Policer Configuration Overview | **2034**

Hierarchical Policer Configuration Overview | **1928**

Policer Color-Marking and Actions | **1968**

bandwidth-limit (Hierarchical Policer)

bandwidth-limit (Policer)

bandwidth-percent

burst-size-limit (Hierarchical Policer)

burst-size-limit (Policer)

Dual Token Bucket Algorithms

IN THIS SECTION

- Token Bucket Concepts | **1974**
- Guaranteed Bandwidth for Three-Color Marking | **1974**
- Nonconformance Measurement for Single-Rate Three-Color Marking | **1974**
- Nonconformance Measurement for Two-Rate Three-Color Marking | **1975**

Token Bucket Concepts

When you apply traffic policing to the input or output traffic at an interface, the rate limits and actions specified in the policer configuration are used to enforce a limit on the average throughput rate at the interface while also allowing bursts of traffic up to a maximum number of bytes based on the overall traffic load. Junos OS policers measure traffic-flow conformance to a policing rate limit by using a *token bucket algorithm*. An algorithm based on a single token bucket allows burst of traffic for short periods, whereas an algorithm based dual token buckets allows more sustained bursts of traffic.

Guaranteed Bandwidth for Three-Color Marking

A committed information rate (CIR) defines the guaranteed bandwidth for traffic arriving at or departing from the interface under normal line conditions. A flow of traffic at an average rate that conforms to the CIR is categorized green, and packets in a green flow are implicitly marked with `low` packet loss priority (PLP) and then passed through the interface. During periods of relatively low traffic (traffic that arrives at or departs from the interface at average rates below the CIR), any unused bandwidth capacity accumulates in the first token bucket, but only up to a configured number of bytes. If any unused bandwidth capacity overflows the first bucket, the excess accumulates in a second token bucket.

The committed burst size (CBS) defines the maximum number of bytes for which unused amounts of the guaranteed bandwidth can be accumulated in the first token bucket. A burst of traffic at an average rate that exceeds the CIR is also categorized as green provided that sufficient unused bandwidth capacity is available in the first token bucket.

Nonconformance Measurement for Single-Rate Three-Color Marking

Single-rate three-color policer configurations specify a second burst size—the excess burst size (EBS)—that defines the maximum number of bytes for which the second token bucket can accumulate unused bandwidth that overflows from the first bucket.

A traffic flow is categorized yellow if its average rate exceeds the CIR and the available bandwidth capacity accumulated in the first bucket if sufficient unused bandwidth capacity is available in the second token bucket. Packets in a yellow flow are implicitly marked with `medium-high` PLP and then passed through the interface.

A traffic flow is categorized red its average rate exceeds the CIR and the available bandwidth capacity accumulated in the second bucket. Packets in a red flow are implicitly marked with `high` PLP and then either passed through the interface or optionally discarded.

Nonconformance Measurement for Two-Rate Three-Color Marking

Two-rate three-color policer configurations include a second rate limit—the peak-information-rate (PIR)—that you set to the expected average data rate for traffic arriving at or departing from the interface under peak conditions.

Two-rate three-color policer configurations also include a second burst size—the peak burst size (PBS)—that defines the maximum number of bytes for which the second token bucket can accumulate unused peak bandwidth capacity. During periods of relatively little peak traffic (traffic that arrives at or departs from the interface at average rates that exceed the PIR), any unused peak bandwidth capacity accumulates in the second token bucket, but only up to the maximum number of bytes specified by the PBS.

A traffic flow is categorized yellow if it exceeds the CIR and the available committed bandwidth capacity accumulated in the first token bucket but conforms to the PIR. Packets in a yellow flow are implicitly marked with `medium-high` PLP and then passed through the interface.

A traffic flow is categorized red if it exceeds the PIR and the available peak bandwidth capacity accumulated in the second token bucket. Packets in a red flow are implicitly marked with `high` PLP and then either passed through the interface or optionally discarded.

RELATED DOCUMENTATION

Three-Color Policer Configuration Overview | **2118**

Policer Color-Marking and Actions | **1968**

committed-burst-size

committed-information-rate

excess-burst-size

peak-burst-size

peak-information-rate

CHAPTER 31

Configuring Layer 2 Policers

IN THIS CHAPTER

- Hierarchical Policers | **1976**
- Configuring a Policer Overhead | **2001**
- Two-Color and Three-Color Policers at Layer 2 | **2003**
- Layer 2 Traffic Policing at the Pseudowire Overview | **2017**
- Configuring a Two-Color Layer 2 Policer for the Pseudowire | **2018**
- Configuring a Three-Color Layer 2 Policer for the Pseudowire | **2019**
- Applying the Policers to Dynamic Profile Interfaces | **2020**
- Attaching Dynamic Profiles to Routing Instances | **2021**
- Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview | **2023**
- Configuring a Policer for the Complex Configuration | **2023**
- Creating a Dynamic Profile for the Complex Configuration | **2024**
- Attaching Dynamic Profiles to Routing Instances for the Complex Configuration | **2026**
- Verifying Layer 2 Traffic Policers on VPLS Connections | **2027**
- Understanding Policers on OVSDB-Managed Interfaces | **2028**
- Example: Applying a Policer to OVSDB-Managed Interfaces | **2029**

Hierarchical Policers

IN THIS SECTION

- Hierarchical Policer Overview | **1977**
- Example: Configuring a Hierarchical Policer | **1978**
- Example: Configuring a Hierarchical Policer for Subscriber Services Firewall (ACX7100-48L Devices) | **1987**

Hierarchical Policer Overview

IN THIS SECTION

- Subscriber Services Firewall Policer (ACX7100-48L Devices) | **1978**

You can use a hierarchical policer to rate-limit ingress Layer 2 traffic at a physical or *logical interface* and apply different policing actions based on whether the packets are classified for expedited forwarding (EF) or for a lower priority.

Hierarchical policing is supported on M40E, M120, and M320 edge routers with incoming Flexible PIC Concentrators (FPCs) as SFPC and outgoing FPCs as FFPC, and on MX Series, ACX7100-L, T320, T640, and T1600 core routers with Enhanced Intelligent Queuing (IQE) PICs.

You can apply hierarchical policing to a logical interface.

A hierarchical policer configuration defines two policers—one for EF traffic only and another for non-EF traffic—that function in a hierarchical manner:

- Premium policer—You configure the premium policer with traffic limits for high-priority EF traffic only: a guaranteed bandwidth and a corresponding burst-size limit. EF traffic is categorized as nonconforming when its average arrival rate exceeds the guaranteed bandwidth and its average packet size exceeds the premium burst-size limit. For a premium policer, the only configurable action for nonconforming traffic is to discard the packets.

- Aggregate policer—You configure the aggregate policer with an aggregate bandwidth (to accommodate both high-priority EF traffic up to the guaranteed bandwidth and normal-priority non-EF traffic) and a burst-size limit for non-EF traffic only. Non-EF traffic is categorized as nonconforming when its average arrival rate exceeds the amount of aggregate bandwidth not currently consumed by EF traffic and its average packet size exceeds the burst-size limit defined in the aggregate policer. For an aggregate policer, the configurable actions for nonconforming traffic are to discard the packets, assign a forwarding class, or assign a packet loss priority (PLP) level.

NOTE: You must configure the bandwidth limit of the premium policer at or below the bandwidth limit of the aggregate policer. If the two bandwidth limits are equal, then non-EF traffic passes through the interface unrestricted only while no EF traffic arrives at the interface.

EF traffic is guaranteed the bandwidth specified as the premium bandwidth limit, while non-EF traffic is rate-limited to the amount of aggregate bandwidth not currently consumed by the EF traffic. Non-EF traffic is rate-limited to the entire aggregate bandwidth only while no EF traffic is present.

For example, suppose that you configure a hierarchical policer with the following components:

- Premium policer with bandwidth limit set to 2 Mbps, burst-size limit set to 3000 bytes, and nonconforming action set to discard packets.
- Aggregate policer with bandwidth limit set to 10 Mbps, burst-size limit set to 3000 bytes, and nonconforming action set to discard packets.

EF traffic is guaranteed a bandwidth of 2 Mbps. Bursts of EF traffic—EF traffic that arrives at the interface at rates above 2 Mbps—can also pass through the interface provided sufficient tokens are available in the 3000-byte bucket. When no tokens are available for a burst of non-EF traffic, packets are rate-limited using policing actions for the premium policer.

Non-EF traffic is metered to a bandwidth limit that ranges between 8 Mbps and 10 Mbps, depending on the average arrival rate of the EF traffic. Bursts of non-EF traffic—non-EF traffic that arrives at the interface at rates above the current limit for non-EF traffic—also pass through the interface provided sufficient tokens are available in the 3000-byte bucket. When non-EF traffic exceeds the currently allowed bandwidth or when no tokens are available for a burst of non-EF traffic, packets are rate-limited using policing actions for the aggregate policer.

Subscriber Services Firewall Policer (ACX7100-48L Devices)

- Starting Junos Evolved release 23.4R1, you can configure the hierarchical policer on DHCP and PPPoE access models at subscriber interfaces, that is activated during subscriber login or CoA. This feature supports colour coded traffic policing according to the colour classification of a packet. The packet colour is classified with packet matching criteria specified for two user-defined traffic classes. See the No Link Title section for related configuration examples and limitations of the feature.

SEE ALSO

Hierarchical Policer Configuration Overview | 1928

Example: Configuring a Hierarchical Policer

No Link Title

Example: Configuring a Hierarchical Policer

IN THIS SECTION

Requirements | 1979

Overview | 1979

Configuration | **1980**
Verification | **1986**

This example shows how to configure a hierarchical policer and apply the policer to ingress Layer 2 traffic at a logical interface on a supported platform.

Requirements

Before you begin, be sure that your environment meets the following requirements:

- The interface on which you apply the hierarchical policer is a SONET interface hosted on one of the following routing platforms:
  - M40e, M120, or M320 edge router with incoming FPCs as SFPC and outgoing FPCs as FFPC.
  - MX Series, T320, T640, or T1600 core router with Enhanced Intelligent Queuing (IQE) PICs.
- No other policer is applied to the input of the interface on which you apply the hierarchical policer.
- You are aware that, if you apply the hierarchical policer to logical interface on which an input filter is also applied, the policer is executed first.

Overview

IN THIS SECTION

Topology | **1979**

In this example, you configure a hierarchical policer and apply the policer to ingress Layer 2 traffic at a logical interface.

Topology

You apply the policer to the SONET logical interface `so-1/0/0.0`, which you configure for IPv4 and VPLS traffic. When you apply the hierarchical policer to that logical interface, both IPv4 and VPLS traffic is hierarchically rate-limited.

You also configure the logical interface so-1/0/0.1 for MPLS traffic. If you choose to apply the hierarchical policer to physical interface so-1/0/0, hierarchical policing would apply to IPv4 and VPLS traffic at so-1/0/0.0 and to MPLS traffic at so-1/0/0.1.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1980**
- Defining the Interfaces | **1981**
- Defining the Forwarding Classes | **1982**
- Configuring the Hierarchical Policer | **1983**
- Applying the Hierarchical Policer to Layer 2 Ingress Traffic at a Physical or Logical Interface | **1984**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the [edit] hierarchy level.

```
set interfaces so-1/0/0 unit 0 family inet address 192.168.1.1/24
set interfaces so-1/0/0 unit 0 family vpls
set interfaces so-1/0/0 unit 1 family mpls
set class-of-service forwarding-classes class fc0 queue-num 0 priority high policing-priority
premium
set class-of-service forwarding-classes class fc1 queue-num 1 priority low policing-priority
normal
set class-of-service forwarding-classes class fc2 queue-num 2 priority low policing-priority
normal
set class-of-service forwarding-classes class fc3 queue-num 3 priority low policing-priority
normal
set firewall hierarchical-policer policer1 aggregate if-exceeding bandwidth-limit 300m burst-
size-limit 30k
set firewall hierarchical-policer policer1 aggregate then forwarding-class fc1
```

```
set firewall hierarchical-policer policer1 premium if-exceeding bandwidth-limit 100m burst-size-
limit 50k
set firewall hierarchical-policer policer1 premium then discard
set interfaces so-1/0/0 unit 0 layer2-policer input-hierarchical-policer policer1
```

Defining the Interfaces

Step-by-Step Procedure

To define the interfaces:

1. Enable configuration of the physical interface.

```
[edit]
user@host# edit interfaces so-1/0/0
```

2. Configure logical unit 0.

```
[edit interfaces so-1/0/0]
user@host# set unit 0 family inet address 192.168.1.1/24
user@host# set unit 0 family vpls
```

   If you apply a Layer 2 policer to this logical interface, you must configure at least one protocol family.

3. Configure logical unit 1.

```
[edit interfaces so-1/0/0]
user@host# set unit 1 family mpls
```

Results

Confirm the configuration of the interfaces by entering the `show interfaces` configuration command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
```

```
        family inet {
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

Defining the Forwarding Classes

Step-by-Step Procedure

To define the forwarding classes referenced as aggregate policer actions:

1. Enable configuration of the forwarding classes.

```
[edit]
user@host# edit class-of-service forwarding-classes
```

2. Define the forwarding classes.

```
[edit class-of-service forwarding-classes]
user@host# set class fc0 queue-num 0 priority high policing-priority premium
user@host# set class fc1 queue-num 1 priority low policing-priority normal
user@host# set class fc2 queue-num 2 priority low policing-priority normal
user@host# set class fc3 queue-num 3 priority low policing-priority normal
```

Results

Confirm the configuration of the forwarding classes referenced as aggregate policer actions by entering the `show class-of-service` configuration command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show class-of-service
forwarding-classes {
    class fc0 queue-num 0 priority high policing-priority premium;
```

```
    class fc1 queue-num 1 priority low policing-priority normal;
    class fc2 queue-num 2 priority low policing-priority normal;
    class fc3 queue-num 3 priority low policing-priority normal;
}
```

Configuring the Hierarchical Policer

Step-by-Step Procedure

To configure a hierarchical policer:

1. Enable configuration of the hierarchical policer.

   ```
   [edit]
   user@host# edit firewall hierarchical-policer policer1
   ```

2. Configure the aggregate policer.

   ```
   [edit firewall hierarchical-policer policer1]
   user@host# set aggregate if-exceeding bandwidth-limit 300m burst-size-limit 30k
   user@host# set aggregate then forwarding-class fc1
   ```

   For the aggregate policer, the configurable actions for a packet in a nonconforming flow are to discard the packet, change the loss priority, or change the forwarding class.

3. Configure the premium policer.

   ```
   [edit firewall hierarchical-policer policer1]
   user@host# set premium if-exceeding bandwidth-limit 100m burst-size-limit 50k
   user@host# set premium then discard
   ```

   The bandwidth limit for the premium policer must not be greater than that of the aggregate policer.

   For the premium policer, the only configurable action for a packet in a nonconforming traffic flow is to discard the packet.

Results

Confirm the configuration of the hierarchical policer by entering the `show firewall` configuration command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
hierarchical-policer policer1 {
    aggregate {
        if-exceeding {
            bandwidth-limit 300m;
            burst-size-limit 30k;
        }
        then {
            forwarding-class fc1;
        }
    }
    premium {
        if-exceeding {
            bandwidth-limit 100m;
            burst-size-limit 50k;
        }
        then {
            discard;
        }
    }
}
```

Applying the Hierarchical Policer to Layer 2 Ingress Traffic at a Physical or Logical Interface

Step-by-Step Procedure

To hierarchically rate-limit Layer 2 ingress traffic for IPv4 and VPLS traffic only on logical interface **so-1/0/0.0**, reference the policer from the logical interface configuration:

1. Enable configuration of the logical interface.

```
[edit]
user@host# edit interfaces so-1/0/0 unit 0
```

When you apply a policer to Layer 2 traffic at a logical interface, you must define at least one protocol family for the logical interface.

2. Apply the policer to the logical interface.

```
[edit]
user@host# set layer2-policer input-hierarchical-policer policer1
```

Alternatively, to hierarchically rate-limit Layer 2 ingress traffic for all protocol families and for *all logical interfaces* configured on physical interface `so-1/0/0`, you could reference the policer from the physical interface configuration.

Results

Confirm the configuration of the hierarchical policer by entering the `show interfaces` configuration command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
        layer2-policer {
            input-hierarchical-policer policer1;
        }
        family inet {
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | 1986
- Displaying Statistics for the Policer | 1986

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interface `so-1/0/0.0`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface, and the **Protocol inet** section contains a **Policer** field that would list the policer `policer1` as an input or output policer as follows:

- **Input: policer1-so-1/0/0.0-inet-i**
- **Output: policer1-so-1/0/0.0-inet-o**

In this example, the policer is applied to logical interface traffic in the input direction only.

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified

policer), in each direction. For the policer `policer1`, the input and output policer names are displayed as follows:

- **policer1-so-1/0/0.0-inet-i**
- **policer1-so-1/0/0.0-inet-o**

The **-inet-i** suffix denotes a policer applied to IPv4 input traffic, while the **-inet-o** suffix denotes a policer applied to IPv4 output traffic. In this example, the policer is applied to input traffic only.

SEE ALSO

Hierarchical Policer Configuration Overview | **1928**

Hierarchical Policer Overview

Example: Configuring a Hierarchical Policer for Subscriber Services Firewall (ACX7100-48L Devices)

IN THIS SECTION

Overview | **1987**

Configuration Scenarios | **1988**

This example shows how to configure a hierarchical policer and apply the policer to ingress traffic at a subscriber interface on a supported platform (ACX7100-48L).

Overview

Starting Junos Evolved release 23.4R1, you can configure hierarchical policers and apply the policers to ingress traffic at a subscriber interface. This feature is supported on ACX7100-48L devices.

NOTE:

- The subscriber services hierarchical policer has the following limitations:

- You can configure only four levels of hierarchical policers with each level having aggregate and premium configuration.
- Logical interface policers and aggregate policers are not supported for subscriber services.
- Policing priority set inside forwarding class is not supported. For example:

```
class-of-service
{
     forwarding-classes
            {
               class BestEffort queue-num 0 priority low policing-priority normal;
            }
}
```

- `next term` is allowed only for hierarchical policers and not for filters.
- Attaching policers directly to an interface or family is not supported. Policers are supported only through firewall filter action.
- Policer action `forwarding-class` is not supported in ingress and egress. `loss-priority high` action is supported in ingress. Only `discard` action is supported in egress.
- A single policy cannot have `loss-priority` and `policer` actions configured. Only one of both actions are supported at any given time.
- `filter-specific` option is not supported in policer configuration.
- Hierarchical policers and tri-color policers are not supported in the Egress direction.
- Filter terms chaining feature is not available.
- Tri-color policer and hierarchical policers are not supported in Egress direction.

Configuration Scenarios

IN THIS SECTION

- Single Rate Two Color Marking Policer Configuration | **1989**
- Single Rate Tri Color Marking Policer Configuration | **1989**
- Two Rate Tri Color Marking Policer | **1993**

- Hierarchical Policer for DHCP and PPPoE | **1995**

The following section provides the different configurations for various hierarchical policer options:

Single Rate Two Color Marking Policer Configuration

Single Rate Tri Color Marking Policer Configuration

You can configure single rate two color marking policer on DHCP and PPPoE access models. The configuration is activated during subscriber login or CoA. View the configuration by entering the `show dynamic-profiles pppoe-client-policer-1-profile` command. A sample configuration is as follows:

```
[edit]

root@bng-controller# show dynamic-profiles pppoe-client-policer-1-profile
variables {
    downstream-inet uid;
    upstream-inet uid;
    HPolicer-in uid;
    HPolicer-out uid;
}
interfaces {
    pp0 {
        unit "$junos-interface-unit" {
            actual-transit-statistics;
            no-traps;
            ppp-options {
                chap;
                pap;
            }
            pppoe-options {
                underlying-interface "$junos-underlying-interface";
                server;
            }
            keepalives interval 30;
            family inet {
                filter {
                    input "$upstream-inet";
                    output "$downstream-inet";
```

```
                }
                unnumbered-address lo0.0;
            }
            family inet6 {
                unnumbered-address lo0.0;
            }
        }
    }
}
firewall
{
    family inet
{
        filter "$downstream-inet"
        {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    policer "$HPolicer-in";
                    accept;
                }
            }
        }
        filter "$upstream-inet" {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    policer "$HPolicer-out";
                    accept;
                }
            }
        }
    }
    policer "$HPolicer-in" {
        if-exceeding {
            bandwidth-limit 5m;
            burst-size-limit 5k;
```

```
        }
        then discard;
    }
    policer "$HPolicer-out" {
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 10k;
        }
        then discard;
    }
}
```

You can configure single rate tri-color marking policer on DHCP and PPPoE access models. It is activated during subscriber login or CoA. View the configuration by entering the `show dynamic-profiles pppoe-client-policer-1-profile` command. A sample configuration is as follows:

```
[edit]
user@bng-controller# show dynamic-profiles pppoe-client-policer-2-profile | no-more
variables
    {
        downstream-inet uid;
        upstream-inet uid;
        HPolicer-in uid;
        HPolicer-out uid;
    }
interfaces
    {
        pp0
            {
                unit "$junos-interface-unit" {
                actual-transit-statistics;
                no-traps;
                ppp-options
                    {
                        chap;
                        pap;
                    }
            pppoe-options
                {
                    underlying-interface "$junos-underlying-interface";
                    server;
                }
```

```
            keepalives interval 30;
            family inet {
                filter {
                    input "$upstream-inet";
                    output "$downstream-inet";
                }
                unnumbered-address lo0.0;
            }
            family inet6 {
                unnumbered-address lo0.0;
            }
        }
    }
}

firewall {
    family inet {
        filter "$downstream-inet" {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    policer "$HPolicer-out";
                    accept;
                }
            }
        }
        filter "$upstream-inet" {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    three-color-policer {
                        single-rate "$HPolicer-in";
                    }
                    count three-color-policer-count;
                    accept;
                }
            }
```

```
        }
    }
policer "$HPolicer-out" {
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 10k;
        }
        then discard;
    }
    three-color-policer "$HPolicer-in" {
        action {
            loss-priority high then discard;
        }
        single-rate {
            color-blind;
            committed-information-rate 5m;
            committed-burst-size 5k;
            excess-burst-size 15k;
        }
    }
}
```

Two Rate Tri Color Marking Policer

You can configure two rate tri color marking policer on DHCP and PPPoE access models. It is activated during subscriber login or CoA.

In this section you can see a sample dynamic profile configuration on controller point. Contents of dynamic profile are propagated forward during subscriber login for a client profile or during service activation for a service profile. View the configuration by entering the `show dynamic-profiles pppoe-client-policer-1-profile` command. A sample configuration is as follows:

```
[edit]

root@bng-controller# show dynamic-profiles pppoe-client-policer-3-profile | no-more
variables {
    downstream-inet uid;
    upstream-inet uid;
    HPolicer-in uid;
    HPolicer-out uid;
}
```

```
interfaces {
    pp0 {
        unit "$junos-interface-unit" {
            actual-transit-statistics;
            no-traps;
            ppp-options {
                chap;
                pap;
            }
            pppoe-options {
                underlying-interface "$junos-underlying-interface";
                server;
            }
            keepalives interval 30;
            family inet {
                filter {
                    input "$upstream-inet";
                    output "$downstream-inet";
                }
                unnumbered-address lo0.0;
            }
            family inet6 {
                unnumbered-address lo0.0;
            }
        }
    }
}
firewall {
    family inet {
        filter "$downstream-inet" {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    policer "$HPolicer-out";
                    accept;
                }
            }
        }
        filter "$upstream-inet" {
            interface-specific;
```

```
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    three-color-policer {
                        two-rate "$HPolicer-in";
                    }
                    count three-color-policer-count;
                    accept;
                }
            }
        }
    }
    policer "$HPolicer-out" {
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 10k;
        }
        then discard;
    }
    three-color-policer "$HPolicer-in" {
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-blind;
            committed-information-rate 5m;
            committed-burst-size 5k;
            peak-information-rate 10m;
            peak-burst-size 10k;
        }
    }
}
```

Hierarchical Policer for DHCP and PPPoE

In this section you can see a sample dynamic profile configuration at the controller point on DHCP and PPPoE access models. Contents of dynamic profile are propagated forward during subscriber login for a

client profile or during service activation for a service profile. View the configuration by entering the `show dynamic-profiles pppoe-client-policer-1-profile` command. A sample configuration is as follows:

```
[edit]
user@host# show dynamic-profiles pppoe-client-policer-4-profile
variables {
    downstream-inet uid;
    upstream-inet uid;
    HPolicer-in uid;
    HPolicer-out uid;
    P0-IN uid;
    P1-IN uid;
    P2-IN uid;
    Session-IN uid;
}
interfaces {
    pp0 {
        unit "$junos-interface-unit" {
            actual-transit-statistics;
            no-traps;
            ppp-options {
                chap;
                pap;
            }
            pppoe-options {
                underlying-interface "$junos-underlying-interface";
                server;
            }
            keepalives interval 30;
            family inet {
                filter {
                    input "$upstream-inet";
                    output "$downstream-inet";
                }
                unnumbered-address lo0.0;
            }
            family inet6 {
                unnumbered-address lo0.0;
            }
        }
    }
}
```

```
firewall {
    family inet {
        filter "$downstream-inet" {
            interface-specific;
            term t1 {
                from {
                    source-port 80;
                }
                then {
                    policer "$HPolicer-out";
                    accept;
                }
            }
        }
        filter "$upstream-inet" {
            interface-specific;
            term P0-Aggregate {
                from {
                    dscp 46;
                }
                then {
                    hierarchical-policer "$P0-IN";
                    next term;
                }
            }
            term P1-Premium {
                from {
                    dscp [ 46 22 ];
                }
                then {
                    force-premium;
                    next term;
                }
            }
            term P1-Aggregate {
                from {
                    dscp [ 46 22 ];
                }
                then {
                    hierarchical-policer "$P1-IN";
                    next term;
                }
            }
```

```
            term P2-Premium {
                from {
                    dscp [ 46 22 56 ];
                }
                then {
                    force-premium;
                    next term;
                }
            }
            term P2-Aggregate {
                from {
                    dscp [ 46 22 56 ];
                }
                then {
                    hierarchical-policer "$P2-IN";
                    next term;
                }
            }
            term final-Premium {
                from {
                    dscp [ 46 22 56 00 ];
                }
                then {
                    force-premium;
                    next term;
                }
            }
            term final {
                then {
                    hierarchical-policer "$Session-IN";
                    accept;
                }
            }
        }
    }
    policer "$HPolicer-out" {
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 10k;
        }
        then discard;
    }
    hierarchical-policer "$HPolicer-in" {
```

```
        aggregate {
            if-exceeding {
                bandwidth-limit 30m;
                burst-size-limit 30k;
            }
            then {
                loss-priority high;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit 10m;
                burst-size-limit 10k;
            }
            then {
                discard;
            }
        }
    }
    hierarchical-policer "$P0-IN" {
        logical-interface-policer;
        aggregate {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
    }
    hierarchical-policer "$P1-IN" {
        logical-interface-policer;
        aggregate {
```

```
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
    }
    hierarchical-policer "$P2-IN" {
        logical-interface-policer;
        aggregate {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
    }
    hierarchical-policer "$Session-IN" {
        logical-interface-policer;
        aggregate {
            if-exceeding {
```

```
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit 5m;
                burst-size-limit 5k;
            }
            then {
                discard;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

SEE ALSO

> Hierarchical Policer Configuration Overview | **1928**
>
> Hierarchical Policer Overview

RELATED DOCUMENTATION

> Hierarchical Policer Configuration Overview | **1928**
>
> Guidelines for Applying Traffic Policers | **1934**

Configuring a Policer Overhead

Configuring a policer overhead allows you to control the rate of traffic sent or received on an interface. When you configure a policer overhead, the configured policer overhead value (bytes) is added to the length of the final Ethernet frame. This calculated length of frame is used to determine the policer or the rate limit action. Therefore, the policer overhead enables you to control the rate of traffic sent or received on an interface. You can configure the policer overhead to rate-limit queues and Layer 2 and

MAC policers. The policer overhead and the shaping overhead can be configured simultaneously on an interface.

This feature is supported on M Series and T Series routers with IQ2 PICs or IQ2E PICs, and on MX Series DPCs.

To configure a policer overhead for controlling the rate of traffic sent or received on an interface:

1. In the `[edit chassis]` hierarchy level in configuration mode, create the interface on which to add the policer overhead to input or output traffic.

```
[edit chassis]
user@host# edit fpc fpc pic pic
```

   For example:

```
[edit chassis]
user@host# edit fpc 0 pic 1
```

2. Configure the policer overhead to control the input or output traffic on the interface. You could use either statement or both the statements for this configuration.

```
[edit chassis fpc fpc pic pic]
user@host# set ingress-policer-overhead bytes;
user@host# set egress-policer-overhead bytes;
```

   For example:

```
[edit chassis fpc 0 pic 1]
user@host# set ingress-policer-overhead 10;
user@host# set egress-policer-overhead 20;
```

3. Verify the configuration:

```
[edit chassis]
user@host# show
fpc 0 {
    pic 1 {
        ingress-policer-overhead 10;
        egress-policer-overhead 20;
```

```
    }
}
```

NOTE: When the configuration for the policer overhead bytes on a PIC is changed, the PIC goes offline and then comes back online. In addition, the configuration in the CLI is on a per-PIC basis and, therefore, applies to all the ports on the PIC.

RELATED DOCUMENTATION

egress-policer-overhead

ingress-policer-overhead

Two-Color and Three-Color Policers at Layer 2

IN THIS SECTION

- Two-Color Policing at Layer 2 Overview | **2003**
- Three-Color Policing at Layer 2 Overview | **2005**
- Example: Configuring a Three-Color Logical Interface (Aggregate) Policer | **2008**

Two-Color Policing at Layer 2 Overview

IN THIS SECTION

- Guidelines for Configuring Two-Color Policing of Layer 2 Traffic | **2004**
- Statement Hierarchy for Configuring a Two-Color Policer for Layer 2 Traffic | **2004**
- Statement Hierarchy for Applying a Two-Color Policer to Layer 2 Traffic | **2005**

Guidelines for Configuring Two-Color Policing of Layer 2 Traffic

The following guidelines apply to two-color policing of Layer 2 traffic:

- You can apply a two-color policer to ingress or egress Layer 2 traffic at a *logical interface* hosted on a Gigabit Ethernet interface (`ge-`) or a 10-Gigabit Ethernet interface (`xe-`) only.
- A single logical interface supports Layer 2 policing in both directions.
- You can apply a two-color policer to Layer 2 traffic as a logical interface policer only. You cannot apply a two-color policer to Layer 2 traffic as a stateless *firewall filter* action.
- You can apply a two-color policer to Layer 2 traffic by referencing the policer in the interface configuration at the logical unit level, and not at the protocol level.

For information about configuring three-color policing of Layer 2 traffic, see Three-Color Policing at Layer 2 Overview.

Statement Hierarchy for Configuring a Two-Color Policer for Layer 2 Traffic

To enable a single-rate two-color policer to rate-limit Layer 2 traffic, include the `logical-interface-policer` statement in the `policer` configuration.

```
firewall {
    policer policer-name {
        logical-interface-policer;
        if-exceeding {
            (bandwidth-limit bps | bandwidth-percent percentage);
            burst-size-limit bytes;
        }
        then {
            discard;
            forwarding-class class-name;
            loss-priority (high | low | medium-high | medium-low);
        }
    }
}
```

You can include the configuration at the following hierarchy levels:

- `[edit]`
- `[edit logical-systems logical-system-name]`

Statement Hierarchy for Applying a Two-Color Policer to Layer 2 Traffic

To apply a logical interface policer to Layer 2 traffic, include the `layer2-policer input-policer` *`policer-name`* statement or the `layer2-policer output-policer` *`policer-name`* statement to a supported logical interface. Use the `input-policer` or `output-policer` statements to apply a two-color policer at Layer 2.

```
interfaces {
    (ge-fpc/pic/port | xe-fpc/pic/port) {
        unit unit-number {
            layer2-policer {
                input-policer policer-name;
                output-policer policer-name;
            }
        }
    }
}
```

You can include the configuration at the following hierarchy levels:

- `[edit]`
- `[edit logical-systems` *`logical-system-name`*`]`

SEE ALSO

> *logical-interface-policer*
>
> *Example: Configuring a Two-Color Logical Interface (Aggregate) Policer*

Three-Color Policing at Layer 2 Overview

IN THIS SECTION

- Guidelines for Configuring Three-Color Policing of Layer 2 Traffic | **2006**
- Statement Hierarchy for Configuring a Three-Color Policer for Layer 2 Traffic | **2006**
- Statement Hierarchy for Applying a Three-Color Policer to Layer 2 Traffic | **2007**

Guidelines for Configuring Three-Color Policing of Layer 2 Traffic

The following guidelines apply to three-color policing of Layer 2 traffic:

- You can apply a three-color policer to Layer 2 traffic at a *logical interface* hosted on a Gigabit Ethernet interface (`ge-`) or a 10-Gigabit Ethernet interface (`xe-`) only.
- A single logical interface supports Layer 2 policing in both directions.
- You can apply a three-color policer to Layer 2 traffic as a logical interface policer only. You cannot apply a two-color policer to Layer 2 traffic as a stateless *firewall filter* action.
- You can apply a three-color policer to Layer 2 traffic by referencing the policer in the interface configuration at the logical unit level, and not at the protocol level.
- You can apply a color-aware three-color policer to Layer 2 traffic in the egress direction only, but you apply a color-blind three-color policer to Layer 2 traffic in either direction.

For information about configuring two-color policing of Layer 2 traffic, see Two-Color Policing at Layer 2 Overview.

Statement Hierarchy for Configuring a Three-Color Policer for Layer 2 Traffic

To enable a single-rate or two-rate three-color policer to rate-limit Layer 2 traffic, include the `logical-interface-policer` statement in the `three-color-policer` configuration.

```
firewall {
    three-color-policer policer-name {
        action {
            loss-priority high then discard;
        }
        logical-interface-policer;
        single-rate {
            (color-aware | color-blind);
            committed-burst-size bytes;
            committed-information-rate bps;
            excess-burst-size bytes;
        }
        two-rate {
            (color-aware | color-blind);
            committed-burst-size bytes;
            committed-information-rate bps;
            peak-burst-size bytes;
            peak-information-rate bps;
```

```
        }
    }
}
```

You can include the configuration at the following hierarchy levels:

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

Statement Hierarchy for Applying a Three-Color Policer to Layer 2 Traffic

To apply a logical interface policer to Layer 2 traffic, include the layer2-policer statement for a supported logical interface at the logical unit level. Use the input-three-color *policer-name* statement or output-three-color *policer-name* statement to specify the direction of the traffic to be policed.

```
interfaces {
    (ge-fpc/pic/port | xe-fpc/pic/port) {
        unit unit-number {
            layer2-policer {
                input-three-color policer-name;
                output-three-color policer-name;
            }
        }
    }
}
```

You can include the configuration at the following hierarchy levels:

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

SEE ALSO

> Example: Configuring a Three-Color Logical Interface (Aggregate) Policer
>
> *layer2-policer*
>
> *logical-interface-policer*
>
> *three-color-policer (Configuring)*

Example: Configuring a Three-Color Logical Interface (Aggregate) Policer

IN THIS SECTION

- Requirements | 2008
- Overview | 2008
- Configuration | 2009
- Verification | 2015

This example shows how to configure a two-rate three-color color-blind policer as a logical interface (aggregate) policer and apply the policer directly to Layer 2 input traffic at a supported logical interface.

Requirements

Before you begin, make sure that the logical interface to which you apply the three-color logical interface policer is hosted on a Gigabit Ethernet interface (`ge-`) or a 10-Gigabit Ethernet interface (`xe-`) on an MX Series router.

Overview

IN THIS SECTION

- Topology | 2009

A two-rate three-color policer meters a traffic flow against a bandwidth limit and burst-size limit for guaranteed traffic, plus a second set of bandwidth and burst-size limits for peak traffic. Traffic that conforms to the limits for guaranteed traffic is categorized as green, and nonconforming traffic falls into one of two categories:

- Nonconforming traffic that does not exceed the bandwidth and burst-size limits for peak traffic is categorized as yellow.
- Nonconforming traffic that exceeds the bandwidth and burst-size limits for peak traffic is categorized as red.

A logical interface policer defines traffic rate-limiting rules that you can apply to multiple protocol families on the same logical interface without creating multiple instances of the policer.

NOTE: You apply a logical interface policer directly to a logical interface at the logical unit level, and not by referencing the policer in a stateless firewall filter and then applying the filter to the logical interface at the protocol family level.

Topology

In this example, you configure the two-rate three-color policer `trTCM2-cb` as a color-blind logical interface policer and apply the policer to incoming Layer 2 traffic on logical interface `ge-1/3/1.0`.

NOTE: When using a three-color policer to rate-limit Layer 2 traffic, color-aware policing can be applied to egress traffic only.

The policer defines guaranteed traffic rate limits such that traffic that conforms to the bandwidth limit of 40 Mbps with a 100 KB allowance for traffic bursting (based on the token-bucket formula) is categorized as green. As with any policed traffic, the packets in a green flow are implicitly set to a `low` loss priority and then transmitted.

Nonconforming traffic that falls within the peak traffic limits of a 60 Mbps bandwidth limit and a 200 KB allowance for traffic bursting (based on the token-bucket formula) is categorized as yellow. The packets in a yellow traffic flow are implicitly set to a `medium-high` loss priority and then transmitted.

Nonconforming traffic that exceeds the peak traffic limits are categorized as red. The packets in a red traffic flow are implicitly set to a `high` loss priority. In this example, the optional policer action for red traffic (`loss-priority high then discard`) is configured, so packets in a red traffic flow are discarded instead of transmitted.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2010**
- Configuring the Logical Interfaces | **2010**
- Configuring the Two-Rate Three-Color Policer as a Logical Interface Policer | **2012**
- Applying the Three-Color Policer to the Layer 2 Input at the Logical Interface | **2014**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces ge-1/3/1 vlan-tagging
set interfaces ge-1/3/1 unit 0 vlan-id 100
set interfaces ge-1/3/1 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/3/1 unit 1 vlan-id 101
set interfaces ge-1/3/1 unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac
00:00:11:22:33:44
set firewall three-color-policer trTCM2-cb logical-interface-policer
set firewall three-color-policer trTCM2-cb two-rate color-blind
set firewall three-color-policer trTCM2-cb two-rate committed-information-rate 40m
set firewall three-color-policer trTCM2-cb two-rate committed-burst-size 100k
set firewall three-color-policer trTCM2-cb two-rate peak-information-rate 60m
set firewall three-color-policer trTCM2-cb two-rate peak-burst-size 200k
set firewall three-color-policer trTCM2-cb action loss-priority high then discard
set interfaces ge-1/3/1 unit 0 layer2-policer input-three-color trTCM2-cb
```

Configuring the Logical Interfaces

Step-by-Step Procedure

To configure the logical interfaces:

1. Enable configuration of the interface.

```
[edit]
user@host# edit interfaces ge-1/3/1
```

2. Configure single tagging.

```
[edit interfaces ge-1/3/1]
user@host# set vlan-tagging
```

3. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 0 vlan-id 100
user@host# set unit 0 family inet address 10.10.10.1/30
```

4. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 1 vlan-id 101
user@host# set unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac 00:00:11:22:33:44
```

Results

Confirm the configuration of the logical interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.10.10.1/30;
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
```

```
            }
        }
    }
}
```

Configuring the Two-Rate Three-Color Policer as a Logical Interface Policer

Step-by-Step Procedure

To configure the two-rate three-color policer as a logical interface policer:

1. Enable configuration of a three-color policer.

```
[edit]
user@host# edit firewall three-color-policer trTCM2-cb
```

2. Specify that the policer is a logical interface (aggregate) policer.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set logical-interface-policer
```

A logical interface policer rate-limits traffic based on a percentage of the media rate of the physical interface underlying the logical interface to which the policer is applied, and the policer is applied directly to the interface rather than referenced by a firewall filter.

3. Specify that the policer is two-rate and color-blind.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate color-blind
```

A color-aware three-color policer takes into account any coloring markings that might have been set for a packet by another traffic policer configured at a previous network node, and any preexisting color markings are used in determining the appropriate policing action for the packet.

Because you are applying this three-color policer applied to input at Layer 2, you must configure the policer to be color-blind.

4. Specify the policer traffic limits used to classify a green traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate committed-information-rate 40m
user@host# set two-rate committed-burst-size 100k
```

5. Specify the additional policer traffic limits used to classify a yellow or red traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate peak-information-rate 60m
user@host# set two-rate peak-burst-size 200k
```

6. (Optional) Specify the configured policer action for packets in a red traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set action loss-priority high then discard
```

In color-aware mode, the three-color policer configured action can increase the packet loss priority (PLP) level of a packet, but never decrease it. For example, if a color-aware three-color policer meters a packet with a medium PLP marking, it can raise the PLP level to high, but cannot reduce the PLP level to low.

Results

Confirm the configuration of the three-color policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
three-color-policer trTCM2-cb {
    logical-interface-policer;
    action {
        loss-priority high then discard;
    }
    two-rate {
        color-blind;
        committed-information-rate 40m;
        committed-burst-size 100k;
```

```
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Applying the Three-Color Policer to the Layer 2 Input at the Logical Interface

Step-by-Step Procedure

To apply the three-color policer to the Layer 2 input at the logical interface:

1. Enable application of Layer 2 logical interface policers.

```
[edit]
user@host# edit interfaces ge-1/3/1 unit 0
```

2. Apply the three-color logical interface policer to a logical interface input.

```
[edit interfaces ge-1/3/1 unit 0]
user@host# set layer2-policerinput-three-color trTCM2-cb
```

Results

Confirm the configuration of the logical interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        layer2-policer {
            input-three-color trTCM2-cb;
        }
        family inet {
            address 10.10.10.1/30;
        }
```

```
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | **2015**
- Displaying Statistics for the Policer | **2016**

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interface `ge-1/3/1.0`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface, and the **Protocol inet** section contains a **Policer** field that would list the policer `trTCM2-cb` as an input or output policer as follows:

- **Input: trTCM2-cb-ge-1/3/1.0-log_int-i**
- **Output: trTCM2-cb-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to in the input direction only.

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction. For the policer `trTCM2-cb`, the input and output policer names are displayed as follows:

- **trTCM2-cb-ge-1/3/1.0-log_int-i**
- **trTCM2-cb-e-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to input traffic only.

SEE ALSO

> Logical Interface (Aggregate) Policer Overview
>
> Example: Configuring a Two-Color Logical Interface (Aggregate) Policer
>
> Three-Color Policing at Layer 2 Overview
>
> *layer2-policer*
>
> *logical-interface-policer*
>
> *three-color-policer (Configuring)*

RELATED DOCUMENTATION

> Guidelines for Applying Traffic Policers | **1934**
>
> *layer2-policer*
>
> *logical-interface-policer*

policer (Configuring)

three-color-policer (Configuring)

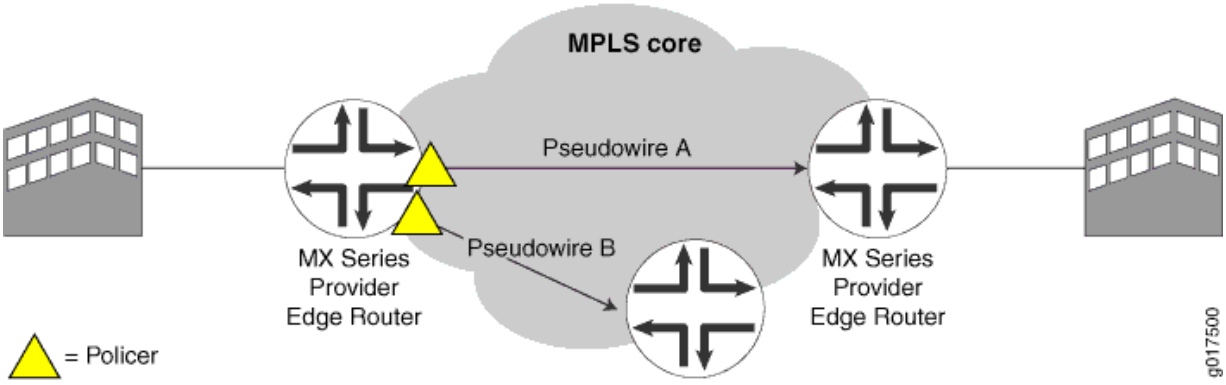
Layer 2 Traffic Policing at the Pseudowire Overview

This feature limits traffic that is sent over the core by policing traffic at the Layer 2 pseudowire level. It uses dynamic profiles to attach two- or three-color policers to pseudowire logical interfaces. You apply the dynamic profiles to core-facing egress interfaces so that they can police unicast, multicast, and broadcast traffic that is going over the MPLS core network.

NOTE: Pseudowire policer statistics collected by the Routing Engine, kernel, and Packet Forwarding Engine can be displayed on the Routing Engine when you execute the `show interfaces` command.

Figure 85 on page 2017 shows an MX Series 5G Universal Routing Platform configured as a provider edge (PE) router. It communicates with other PE routers over pseudowires. It can aggregate both unicast and multicast traffic and send it over pseudowires. To limit traffic over the pseudowires, you can set up policers on each pseudowire that faces the MPLS core network.

Figure 85: Limiting Traffic to the Core Using Layer 2 Policers at the Pseudowire Level



NOTE: This feature is supported only on pseudowire logical interfaces at the egress. It is not supported on tunnel interfaces.

Configuring a Two-Color Layer 2 Policer for the Pseudowire

For the basic configuration of Layer 2 policers for pseudowires, create a two-color policer.

To configure a two-color policer:

1. Create a two-color policer.

```
[edit firewall]
user@host#  edit policer 2color-l2-policer
```

2. Specify that the policer is to be used on a logical interface.

```
[edit firewall policer 2color-l2-policer]
user@host#  set logical-interface-policer
```

3. Configure the policer.

```
[edit firewall policer 2color-l2-policer]
user@host#  edit if-exceeding
[edit firewall policer 2color-l2-policer if-exceeding]
user@host#  set bandwidth-limit 30m
user@host#  set burst-size-limit 300k
```

4. Set the action that the policer takes to loss-priority and specify that the packet loss priority (PLP) is high.

```
[edit firewall policer 2color-l2-policer]
user@host#  set then loss-priority high
```

RELATED DOCUMENTATION

Layer 2 Traffic Policing at the Pseudowire Overview | 2017

Configuring a Three-Color Layer 2 Policer for the Pseudowire | 2019

Applying the Policers to Dynamic Profile Interfaces | 2020

Attaching Dynamic Profiles to Routing Instances | 2021

Configuring a Three-Color Layer 2 Policer for the Pseudowire

For the basic configuration of Layer 2 policers for pseudowires, create a three-color policer. This scenario shows a two-rate three-color-marking (trTCM) policer.

To configure a three-color policer:

1. Create a three-color policer.

```
[edit firewall]
user@host# edit three-color-policer trTCM-policer
```

2. Specify that the policer is to be used on a logical interface.

```
[edit firewall three-color-policer trTCM-policer]
user@host# set logical-interface-policer
```

3. Set the action for the policer.

```
[edit firewall three-color-policer trTCM-policer]
user@host# set action loss-priority high then discard
```

4. Specify that the policer is a two-rate policer and configure the policer.

```
[edit firewall three-color-policer trTCM-policer]
user@host# edit two-rate
user@host# set color-aware
user@host# set committed-information-rate 10m
user@host# set committed-burst-size 50m
user@host# set committed-burst-size 150k
user@host# set peak-information-rate 50m
user@host# set peak-burst-size 450k
```

RELATED DOCUMENTATION

Layer 2 Traffic Policing at the Pseudowire Overview | 2017

Two-Rate Three-Color Policer Overview | 2147

Configuring a Two-Color Layer 2 Policer for the Pseudowire | 2018

Applying the Policers to Dynamic Profile Interfaces | 2020
Attaching Dynamic Profiles to Routing Instances | 2021

Applying the Policers to Dynamic Profile Interfaces

This configuration shows how to apply policers to a dynamic profile.

Before you can apply policers, you need to have configured your policers as described in:

- "Configuring a Three-Color Layer 2 Policer for the Pseudowire" on page 2019
- "Configuring a Two-Color Layer 2 Policer for the Pseudowire" on page 2018

To configure the dynamic profiles:

1. Create a dynamic profile for the three-color policer.

```
[edit dynamic-profiles]
user@host#  edit pw-trTCM-policer
```

2. Create a dynamic profile interface that has a dynamic underlying interface unit.

```
[edit dynamic-profiles pw-trTCM-policer]
user@host#  edit interfaces $junos-interface-ifd-name unit $junos-underlying-interface-unit
```

3. Specify that VPLS is the protocol family.

```
[edit dynamic-profiles pw-trTCM-policer interfaces "$junos-interface-ifd-name" unit "$junos-
underlying-interface-unit"]
user@host#  set family vpls
```

4. Assign the three-color policer to the dynamic profile.

```
[edit dynamic-profiles pw-trTCM-policer interfaces "$junos-interface-ifd-name" unit "$junos-
underlying-interface-unit"]
user@host#  set layer2-policer output-three-color trTCM-policer
```

5. Create a dynamic profile for the two-color policer.

```
[edit dynamic-profiles]
user@host#  edit pw-2color-l2-policer
```

6. Create a dynamic profile interface that has a dynamic underlying interface unit.

```
[edit dynamic-profiles pw-2color-l2-policer]
user@host#  edit interfaces $junos-interface-ifd-name unit $junos-underlying-interface-unit
```

7. Specify that VPLS is the protocol family.

```
[edit dynamic-profiles pw-2color-l2-policer interfaces "$junos-interface-ifd-name" unit
"$junos-underlying-interface-unit"]
user@host#  set family vpls
```

8. Assign the two-color policer to the dynamic profile.

```
[edit dynamic-profiles pw-2color-l2-policer interfaces "$junos-interface-ifd-name" unit
"$junos-underlying-interface-unit"]
user@host#  set layer2-policer output-policer 2color-l2-policer
```

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | **2017**
- Configuring a Three-Color Layer 2 Policer for the Pseudowire | **2019**
- Configuring a Two-Color Layer 2 Policer for the Pseudowire | **2018**
- Attaching Dynamic Profiles to Routing Instances | **2021**

Attaching Dynamic Profiles to Routing Instances

To bind the dynamic profile to the pseudowire, attach it to a routing instance. The routing instance can be a VPLS instance type or a virtual switch instance type. You can attach dynamic profiles to the routing instance at the VPLS protocol level, at the mesh-group level, or at the neighbor level.

Because this feature is not supported on tunnel interfaces, for VPLS routing interfaces, you must include the `no-tunnel-services` statement at the `[edit routing-instances routing-instance-name protocols vpls]` hierarchy level.

- To attach the dynamic profile at the VPLS protocol level:

```
[edit routing-instances]
user@host#  edit green protocols vpls associate-profile
[edit routing-instances green protocols vpls associate-profile]
user@host#  set pw-2color-l2-policer
```

- To attach the dynamic profile at the mesh-group level:

```
[edit routing-instances]
user@host#  edit green protocols vpls mesh-group lata-1 associate-profile
[edit routing-instances green protocols vpls mesh-group lata-1 associate-profile]
user@host#  set pw-trTCM-policer
```

- To attach the dynamic profile at the neighbor level:

```
[edit routing-instances]
user@host#  edit green protocols vpls mesh-group lata-1 neighbor 10.10.1.1 associate-profile
[edit routing-instances green protocols vpls mesh-group lata-1 neighbor 10.10.1.1 associate-
profile]
user@host#  set pw-2color-l2-policer
```

RELATED DOCUMENTATION

Layer 2 Traffic Policing at the Pseudowire Overview | 2017

Configuring a Three-Color Layer 2 Policer for the Pseudowire | 2019

Configuring a Two-Color Layer 2 Policer for the Pseudowire | 2018

Applying the Policers to Dynamic Profile Interfaces | 2020

Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview

To reduce the number of dynamic profiles needed to police traffic at the core, you can use a variable for the output policer in your dynamic profiles. The variable that you define is called **junos-layer2-output-policer**. The variable is a placeholder that gets filled in when the dynamic profile obtains the value from the routing instance.

To use variables for policers for Layer 2 pseudowires:

1. Create policers.
2. Create a dynamic profile and add a profile variable set to the dynamic profile.
3. In the profile variable set, assign a value to the **junos-layer2-output-policer** variable. This value is the name of one of your policers.
4. In the dynamic profile interface configuration at the `[edit dynamic-profiles` *`profile-name`* `interfaces "$junos-interface-ifd-name" unit "$junos-underlying-interface-unit"]` hierarchy, assign **junos-layer2-output-policer** as one of your Layer 2 policers.
5. When you attach the dynamic profile to a routing instance, assign the profile variable set that you configured in the dynamic profile as the **associate-profile** value.
6. Attach the dynamic profile and the profile variable set to the routing instance.

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | **2017**
- Configuring a Policer for the Complex Configuration | **2023**
- Creating a Dynamic Profile for the Complex Configuration | **2024**
- Attaching Dynamic Profiles to Routing Instances for the Complex Configuration | **2026**

Configuring a Policer for the Complex Configuration

For the complex configuration of Layer 2 policers for pseudowires, create a two-color policer.

To configure a two-color policer:

1. Create a two-color policer.

```
[edit firewall]
user@host#  edit policer 10m-policer
```

2. Specify that the policer is to be used on a logical interface.

```
[edit firewall policer 10m-policer]
user@host#  set logical-interface-policer
```

3. Configure the policer.

```
[edit firewall policer 10m-policer]
user@host#  edit if-exceeding
[edit firewall policer 10m-policer if-exceeding]
user@host#  set bandwidth-limit 10m
user@host#  set burst-size-limit 100k
```

4. Set the action that the policer takes to loss-priority and specify that the packet loss priority (PLP) is high.

```
[edit firewall policer 10m-policer]
user@host#  set then loss-priority high
```

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | 2017
- Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview | 2023
- Creating a Dynamic Profile for the Complex Configuration | 2024
- Attaching Dynamic Profiles to Routing Instances for the Complex Configuration | 2026

Creating a Dynamic Profile for the Complex Configuration

For this configuration, the dynamic profile defines a profile variable set and then assigns the variable to the output policer.

To configure dynamic profiles:

1. Create a dynamic profile.

```
[edit  dynamic-profiles]
user@host#  edit pw-policer
```

2. Create a profile variable set and define the **junos-layer2-output-policer** variable. In this scenario, set the variable to the **10m-policer**.

```
[edit dynamic-profiles pw-policer]
user@host#  edit profile-variable-set pw-policer-var-set
user@host#  set junos-layer2-output-policer 10m-policer
```

3. Create a dynamic profile interface that has a dynamic underlying interface unit.

```
[edit dynamic-profiles pw-policer]
user@host#  edit interfaces $junos-interface-ifd-name unit $junos-underlying-interface-unit
```

4. Assign the **junos-layer2-output-policer** variable to the two-color output policer.

```
[edit dynamic-profiles pw-policer interfaces "$junos-interface-ifd-name" unit "$junos-
underlying-interface-unit"]
user@host#  set layer2-policer output-policer $junos-layer2-output-policer
```

5. Specify that VPLS is the protocol family.

```
[edit dynamic-profiles pw-2color-l2-policer interfaces "$junos-interface-ifd-name" unit
"$junos-underlying-interface-unit"]
user@host#  set family vpls
```

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | **2017**
- Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview | **2023**
- Configuring a Policer for the Complex Configuration | **2023**
- Attaching Dynamic Profiles to Routing Instances for the Complex Configuration | **2026**

Attaching Dynamic Profiles to Routing Instances for the Complex Configuration

To bind the dynamic profile to the pseudowire, attach it to a routing instance. When your dynamic profile contains variables, you assign one of the profile variable sets that you configured in your dynamic profile when you associate the profile with the routing instance.

The routing instance can be a VPLS instance type or a virtual switch instance type. You can attach the dynamic profile and the profile variable set to the routing instance at the VPLS protocol level, at the mesh-group level, or at the neighbor level.

Because this feature is not supported on tunnel interfaces, for VPLS routing interfaces, you must include the `no-tunnel-services` statement at the `[edit routing-instances` *routing-instance-name* `protocols vpls]` hierarchy level.

- To attach the dynamic profile and the profile variable set at the VPLS protocol level:

```
[edit routing-instances]
user@host#  edit green protocols vpls associate-profile
[edit routing-instances green protocols vpls associate-profile]
user@host#  set profile-variable-set pw-policer
user@host#  set profile-variable-set pw-policer-var-set
```

- To attach the dynamic profile and the profile variable set at the mesh-group level:

```
[edit routing-instances]
user@host#  edit green protocols vpls mesh-group lata-1 associate-profile
[edit routing-instances green protocols vpls mesh-group lata-1 associate-profile]
user@host#  set profile-variable-set pw-policer
user@host#  setprofile-variable-set pw-policer-var-set
```

- To attach the dynamic profile and the profile variable set at the neighbor level:

```
[edit routing-instances]
user@host#  edit green protocols vpls mesh-group lata-1 neighbor 10.10.1.1 associate-profile
[edit routing-instances green protocols vpls mesh-group lata-1 neighbor 10.10.1.1 associate-
profile]
user@host#  profile-variable-set pw-policer
user@host#  profile-variable-set pw-policer-var-set
```

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | 2017
- Using Variables for Layer 2 Traffic Policing at the Pseudowire Overview | 2023
- Configuring a Policer for the Complex Configuration | 2023
- Creating a Dynamic Profile for the Complex Configuration | 2024

Verifying Layer 2 Traffic Policers on VPLS Connections

IN THIS SECTION

- Purpose | 2027
- Action | 2027
- Meaning | 2028

Purpose

Display VPLS connections to verify that the dynamic profile is running on the Layer 2 VPN connection.

Action

```
user@host> show vpls connections
Layer-2 VPN connections:

...
Instance: vpls-10gige
  Local site: 10Gige-PE (2)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Mar 28 21:27:57 2010           1
      Remote PE: 10.10.1.1, Negotiated control-word: No
      Incoming label: 262145, Outgoing label: 262146
      Local interface: lsi.1048576, Status: Up, Encapsulation: VPLS
      Dynamic profile: pw-policer
        Description: Intf - vpls vpls-10gige local site 2 remote site 1
```

Meaning

The Dynamic profile field displays the policer that is currently running on the VPLS connection.

RELATED DOCUMENTATION

- Layer 2 Traffic Policing at the Pseudowire Overview | **2017**

Understanding Policers on OVSDB-Managed Interfaces

When you use a Contrail controller to manage VXLANs on a QFX switch (through the Open vSwitch Database—OVSDB—management protocol), the VXLAN interfaces are automatically configured with the `flexible-vlan-tagging` and `encapsulation extended-vlan-bridge` statements. Starting with Junos OS Release 14.1X53-D30, you can create `family ethernet-switching` logical units (subinterfaces) on VXLAN interfaces. This enables you to apply firewall filters with the action `three-color-policer` to these subinterfaces, which means that you can apply two-rate three-color markers (policers) to OVSDB-managed interfaces. See "Example: Applying a Policer to OVSDB-Managed Interfaces" on page 2029 for information about how to configure policers on VXLAN interfaces managed by a Contrail controller.

> **NOTE**: Firewall filters are the only supported configuration items on `family ethernet-switching` subinterfaces of OVSDB-managed interfaces. Two-rate three-color markers are the only supported policers.

Release History Table

Release	Description
14.1X53-D30	Starting with Junos OS Release 14.1X53-D30, you can create `family ethernet-switching` logical units (subinterfaces) on VXLAN interfaces.

RELATED DOCUMENTATION

- Example: Applying a Policer to OVSDB-Managed Interfaces | **2029**
- Overview of Policers | **2198**
- *Understanding VXLANs*
- *Understanding the OVSDB Protocol Running on Juniper Networks Devices*

Understanding Firewall Filters on OVSDB-Managed Interfaces | 1488

Example: Applying a Policer to OVSDB-Managed Interfaces

IN THIS SECTION

- Requirements | 2029
- Overview | 2030
- Configuration | 2030

Starting with Junos OS Release 14.1X53-D30, you can create `family ethernet-switching` logical units (subinterfaces) on VXLAN interfaces managed by a Contrail controller. (The controller and switch communicate through the Open vSwitch Database—OVSDB—management protocol). This support enables you to apply firewall filters with the action `three-color-policer` to these subinterfaces, which means that you can apply two-rate three-color markers (policers) to OVSDB-managed interfaces.

Because a Contrail controller can create subinterfaces dynamically, you need to apply firewall filters in such a way that the filters will apply to subinterfaces whenever the controller creates them. You accomplish this by using configuration groups to configure and apply the firewall filters. (You must use configuration groups for this purpose—that is, you cannot apply a firewall filter directly to these subinterfaces.)

NOTE: Firewall filters are the only supported configuration items on `family ethernet-switching` subinterfaces of OVSDB-managed interfaces. Two-rate three-color markers are the only supported policers.

Requirements

This example uses the following hardware and software components:

- A QFX5100 switch
- Junos OS Release 14.1X53-D30 or later

Overview

This example assumes that interfaces xe-0/0/0 and xe-0/0/1 on the switch are VXLAN interfaces managed by a Contrail controller, which means that the controller has applied the `flexible-vlan-tagging` and `encapsulation extended-vlan-bridge` statements to these interfaces. To apply a firewall filter Layer 2 (port) firewall filter with a policer action to any subinterfaces that the controller creates dynamically, you must create and apply the filter as shown in this example.

NOTE: As shown in the example, all of the statements must be part of a configuration group when you want to apply a firewall filter (and policer) to an OVSDB-managed subinterface.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 2030
- Procedure | 2031

To configure a firewall filter with a policer action to be automatically applied to subinterfaces created dynamically by a Contrail controller, perform these tasks:

CLI Quick Configuration

```
[edit]
set groups vxlan-policer-group interfaces xe-0/0/0 unit <*> family ethernet-switching filter
input vxlan-filter
set groups vxlan-policer-group interfaces xe-0/0/1 unit <*> family ethernet-switching filter
input vxlan-filter
set groups vxlan-policer-group firewall three-color-policer vxlan-policer action loss-priority
high then discard
set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate color-blind
set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate committed-
burst-size 2m
set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate committed-
information-rate 100m
set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate peak-burst-
```

```
size 4m
set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate peak-
information-rate 100m
set groups vxlan-policer-group firewall family ethernet-switching filter vxlan-filter term t1
then three-color-policer two-rate vxlan-policer
set apply-groups vxlan-policer-group
```

Procedure

Step-by-Step Procedure

1. Create configuration group `vxlan-policer-group` to apply firewall filter `vxlan-filter` to any subinterface of interface xe-0/0/0. The filter applies to any subinterface because you specify `unit <*>`:

   ```
   [edit]
   user@switch# set groups vxlan-policer-group interfaces xe-0/0/0 unit <*> family ethernet-
   switching filter input vxlan-filter
   ```

2. Create the same configuration for interface xe-0/0/1:

   ```
   [edit]
   user@switch# set groups vxlan-policer-group interfaces xe-0/0/1 unit <*> family ethernet-
   switching filter input vxlan-filter
   ```

3. Configure the policer to discard packets with high loss priority. (Junos OS assigns high loss priority to packets that exceed the peak information rate and the peak burst size.) As with the interface configuration, you must also configure the policer to be part of a configuration group.

   ```
   [edit]
   user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer
   action loss-priority high then discard
   ```

4. Configure the policer to be color blind, which means that it ignores any preclassification of packets and can assign a higher or lower packet loss priority.

```
[edit]
user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate color-blind
```

5. Configure the policer to allow incoming traffic to burst a maximum of 2 megabytes above the committed information rate and still be marked with low packet loss priority (green).

```
[edit]
user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate committed-burst-size 2m
```

6. Configure the policer to allow guaranteed bandwidth of 100 megabytes under normal line conditions. This is the average rate up threshold under which packets are marked with low packet loss priority (green).

```
[edit]
user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate committed-information-rate 100m
```

7. Configure the policer to allow incoming packets to burst a maximum of 4 megabytes above the peak information rate and still be marked with medium-high packet loss priority (yellow). Packets that exceed the peak burst size are marked with high packet loss priority (red).

```
[edit]
user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate peak-burst-size 4m
```

8. Configure the policer to allow a maximum achievable rate of 100 megabytes. Packets that exceed the committed information rate but are below the peak information rate are marked with medium-

high packet loss priority (yellow). Packets that exceed the peak information rate are marked with high packet loss priority (red).

```
[edit]
user@switch# set groups vxlan-policer-group firewall three-color-policer vxlan-policer two-rate peak-information-rate 100m
```

9. Configure the firewall filter `vxlan-filter` to send matching packets (all packets, because there is no `from` statement) to the policer:

```
[edit]
user@switch# set groups vxlan-policer-group firewall family ethernet-switching filter vxlan-filter term t1 then three-color-policer two-rate vxlan-policer
```

10. Apply the group to enable its configuration:

```
[edit]
user@switch# set apply-groups vxlan-policer-group
```

RELATED DOCUMENTATION

- *Understanding Junos OS Configuration Groups*
- Overview of Firewall Filters (QFX Series) | **1716**
- Overview of Policers | **2198**
- *Understanding VXLANs*
- *Understanding the OVSDB Protocol Running on Juniper Networks Devices*
- Understanding Policers on OVSDB-Managed Interfaces | **2028**

CHAPTER 32

Configuring Two-Color and Three-Color Traffic Policers at Layer 3

IN THIS CHAPTER

- Two-Color Policer Configuration Overview | **2034**
- Basic Single-Rate Two-Color Policers | **2042**
- Bandwidth Policers | **2070**
- Prefix-Specific Counting and Policing Actions | **2083**
- Policer Overhead to Account for Rate Shaping in the Traffic Manager | **2106**
- Three-Color Policer Configuration Overview | **2118**
- Applying Policers | **2122**
- Three-Color Policer Configuration Guidelines | **2134**
- Basic Single-Rate Three-Color Policers | **2138**
- Basic Two-Rate Three-Color Policers | **2147**
- Example: Configuring a Two-Rate Three-Color Policer | **2157**

Two-Color Policer Configuration Overview

Table 115 on page 2035 describes the hierarchy levels at which you can configure and apply single-rate two-color policers to Layer 3 traffic. For information about applying single-rate two-color policers to Layer 2 traffic, see "Two-Color Policing at Layer 2 Overview" on page 2003.

Table 115: Two-Color Policer Configuration and Application Overview

Policer Configuration	Layer 3 Application	Key Points
Single-Rate Two-Color Policer *Defines traffic rate limiting that you can apply to Layer 3 protocol-specific traffic at a logical interface. Can be applied as an interface policer or as a firewall filter policer.*		

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Basic policer configuration:
 	
`[edit firewall]`
`policer policer-name {`
`  if-exceeding {`
`  bandwidth-limit bps;`
`  burst-size-limit bytes;`
`  }`
`  then {`
`  discard;`
`  forwarding-class class-name;`
`  loss-priority supported-value;`
`  }`
`}`
	Method A—Apply as an interface policer at the protocol family level:
	
	`[edit interfaces]`
	`interface-name {`
	`  unit unit-number {`
	`  family family-name {`
	`  policer {`
	`  input policer-name;`
	`  output policer-name;`
	`  }`
	`  }`
	`  }`
	`}`
	
	Method B—Apply as a firewall filter policer at the protocol family level:
	
	`[edit firewall]`
	`family family-name {`
	`  filter filter-name {`
	`  interface-specific; # (*)`
	`  from {`
	`  ... match-conditions ...`
	`  }`
	`  then {`
	`  policer policer-name;`
	`  }`
	`  }`
	`}`
	
	`[edit interfaces]`
	`interface-name {`
	`  unit unit-number {`
	`  family family-name {`
	`  filter {`
	`  input filter-name;`
	`  output filter-name;`
	`  }`
	`  ... protocol-configuration ...`
	` }
		Policer configuration:
		
		• Use **bandwidth-limit *bps*** to specify an absolute value.
		
		Firewall filter configuration (*)
		
		• If applying to multiple interfaces, include the **interface-specific** statement to create unique policers and counters for each interface.
		
		Interface policer verification:
		
		• Use the **show interfaces (detail | extensive)** operational mode command.
		
		• Use the **show policer** operational mode command.
		
		Firewall filter policer verification:
		
		• Use the **show interfaces (detail | extensive)** operational mode command.
		
		• Use the **show firewall filter *filter-name*** operational mode command.

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points
	} }	

Bandwidth Policer

Defines traffic rate limiting that you can apply to Layer 3 protocol-specific traffic at a logical interface, but the bandwidth limit is specified as a percentage value. Bandwidth can be based on physical interface line rate (the default) or the logical interface shaping rate. Can be applied as an interface policer or as a firewall filter policer where the filter is either interface-specific or a physical interface filter.

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Bandwidth policer configuration:
 	
`[edit firewall]`
`policer policer-name {`
`  logical-bandwidth-policer;`
`  if-exceeding {`
`  bandwidth-percent (1..100);`
`  burst-size-limit bytes;`
`  }`
`  then {`
`  discard;`
`  forwarding-class class-name;`
`  loss-priority supported-value;`
`  }`
`}`
	Method A—Apply as an interface policer at the protocol family level:
	
	`[edit interfaces]`
	`interface-name {`
	`  unit unit-number {`
	`  family family-name {`
	`  policer {`
	`  input policer-name;`
	`  output policer-name;`
	`  }`
	`  }`
	`  }`
	`}`
	
	Method B—Apply as a firewall filter policer at the protocol family level:
	
	`[edit firewall]`
	`family family-name {`
	`  filter filter-name {`
	`  interface-specific;`
	`  from {`
	`  ... match-conditions ...`
	`  }`
	`  then {`
	`  policer policer-name;`
	`  }`
	`  }`
	`}`
	
	`[edit interfaces]`
	`interface-name {`
	`  unit unit-number {`
	`  family family-name {`
	`  filter {`
	`  input filter-name;`
	`  output filter-name;`
	`  }`
	`  ... protocol-configuration ...`
	` }
		Policer configuration:
		
		• Use the **bandwidth-percent *percentage*** statement instead of the **bandwidth-limit *bps*** statement.
		
		By default, bandwidth policing rate-limits traffic based on a percentage of the physical interface media rate.
		
		• To rate-limit traffic based on a percentage of the logical interface configured shaping rate, also include the **logical-bandwidth-policer** statement.
		
		Firewall filter configuration:
		
		• Percentage bandwidth policers can only be referenced by filters configured with the **interface-specific** statement.
		
		Interface policer verification:
		
		• Use the **show interfaces (detail | extensive)** operational mode command.
		
		• Use the **show policer** operational mode command.
		
		Firewall filter policer verification:

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points
	} }	• Use the **show interfaces (detail | extensive)** operational mode command. • Use the **show firewall filter *filter-name*** operational mode command.
Logical Interface (Aggregate) Policer *Defines traffic rate limiting that you can apply to multiple protocol families on the same logical interface without creating multiple instances of the policer. Can be applied directly to a logical interface configuration only.*		

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Logical interface policer configuration:
 	
`[edit firewall]`
`policer` *`policer-name`* `{`
`  logical-interface-policer;`
`  if-exceeding {`
`  bandwidth-limit` *`bps`*`;`
`  burst-size-limit` *`bytes`*`;`
`  }`
`  then {`
`  discard;`
`  forwarding-class` *`class-name`*`;`
`  loss-priority` *`supported-value`*`;`
`  }`
`}`
	Apply as an interface policer only:
	
	`[edit interfaces]`
	`interface-name` `{`
	`  unit` *`unit-number`* `{`
	`  policer { # All protocols`
	`  input` *`policer-name`*`;`
	`  output` *`policer-name`*`;`
	`  }`
	`  family` *`family-name`* `{`
	`  policer { # One protocol`
	`  input` *`policer-name`*`;`
	`  output` *`policer-name`*`;`
	`  }`
	`  }`
	`  }`
	`}`
		Policer configuration:
		
		• Include the **logical-interface-policer** statement.
		
		Two options for interface policer application:
		
		• To rate-limit all traffic types, regardless of the protocol family, apply the logical interface policer at the logical unit level.
		
		• To rate-limit traffic of a specific protocol family, apply the logical interface policer at the protocol family level.
		
		Interface policer verification:
		
		• Use the **show interfaces (detail | extensive)** operational mode command.
		
		• Use the **show policer** operational mode command.

Physical Interface Policer Defines traffic rate limiting that applies to all logical interfaces and protocol families configured on a physical interface, even if the interfaces belong to different routing instances. Can be applied as a firewall filter policer referenced from a physical interface filter only.		

Table 115: Two-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Physical interface policer configuration:
 	
`[edit firewall]`
`policer policer-name {`
`  physical-interface-policer;`
`  if-exceeding {`
`  bandwidth-limit bps;`
`  burst-size-limit bytes;`
`  }`
`  then {`
`  discard;`
`  forwarding-class class-name;`
`  loss-priority supported-value;`
`  }`
`}`
	Apply as a firewall filter policer referenced from a physical interface filter that you apply at the protocol family level:
	
	`[edit firewall]`
	`family family-name {`
	`  filter filter-name {`
	`  physical-interface-filter;`
	`  from {`
	`  ... match-conditions ...`
	`  }`
	`  then {`
	`  policer policer-name;`
	`  }`
	`  }`
	`}`
	
	`[edit interfaces]`
	`interface-name {`
	`  unit number {`
	`  family family-name {`
	`  filter {`
	`  input filter-name;`
	`  output filter-name;`
	`  }`
	`  ... protocol-configuration ...`
	`  }`
	`  }`
	`}`
		Policer configuration:
		
		• Include the **physical-interface-policer** statement.
		
		Firewall filter configuration:
		
		• Include the **physical-interface-filter** statement.
		
		Application:
		
		• Apply the filter to the input or output of a logical interface at the protocol family level.
		
		Firewall filter policer verification:
		
		• Use the **show interfaces (detail | extensive)** operational mode command.
		
		• Use the **show firewall filter *filter-name*** operational mode command.

RELATED DOCUMENTATION

Basic Single-Rate Two-Color Policers | 2042

Bandwidth Policers | 2070

Prefix-Specific Counting and Policing Actions | 2083

Multifield Classifier Example: Configuring Multifield Classification | 1304

Policer Overhead to Account for Rate Shaping in the Traffic Manager | 2106

Two-Color and Three-Color Physical Interface Policers | 2185

Basic Single-Rate Two-Color Policers

IN THIS SECTION

- Single-Rate Two-Color Policer Overview | 2042
- Example: Limiting Inbound Traffic at Your Network Border by Configuring an Ingress Single-Rate Two-Color Policer | 2043
- Example: Configuring Interface and Firewall Filter Policers at the Same Interface | 2055

Single-Rate Two-Color Policer Overview

Single-rate two color policing enforces a configured rate of traffic flow for a particular service level by applying implicit or configured actions to traffic that does not conform to the limits. When you apply a single-rate two-color policer to the input or output traffic at an interface, the policer meters the traffic flow to the rate limit defined by the following components:

- Bandwidth limit—The average number of bits per second permitted for packets received or transmitted at the interface. You can specify the bandwidth limit as an absolute number of bits per second or as a percentage value from 1 through 100. If a percentage value is specified, the effective bandwidth limit is calculated as a percentage of either the physical interface media rate or the *logical interface* configured shaping rate.
- Packets per second (pps) limit (MX Series with MPC only)–The average number of packets per second permitted for packets received or transmitted at the interface. You specify the pps limit as an absolute number of packets per second.
- Burst-size limit—The maximum size permitted for bursts of data.
- Packet burst limit–

For a traffic flow that conforms to the configured limits (categorized as green traffic), packets are implicitly marked with a packet loss priority (PLP) level of `low` and are allowed to pass through the interface unrestricted.

For a traffic flow that exceeds the configured limits (categorized as red traffic), packets are handled according to the traffic-policing actions configured for the policer. The action might be to discard the

packet, or the action might be to re-mark the packet with a specified forwarding class, a specified PLP, or both, and then transmit the packet.

To rate-limit Layer 3 traffic, you can apply a two-color policer in the following ways:

- Directly to a logical interface, at a specific protocol level.
- As the action of a standard stateless *firewall filter* that is applied to a logical interface, at a specific protocol level.

To rate-limit Layer 2 traffic, you can apply a two-color policer as a *logical interface policer* only. You cannot apply a two-color policer to Layer 2 traffic through a firewall filter.

NOTE: On MX platforms, Packet Loss Priority (PLP) is not implicitly to low (green) when the traffic flow confirms to the configured policer limit. Instead it takes the user configured PLP values like high, medium-high, medium-low. Use `dp-rewrite` under `edit firewall policer <policer-name>` to enable this behavior on MX platforms. If the knob is not enabled, then the packets may carry their original color and loss priority.

SEE ALSO

Two-Color Policer Configuration Overview | **2034**

Example: Limiting Inbound Traffic at Your Network Border by Configuring an Ingress Single-Rate Two-Color Policer

Example: Configuring Interface and Firewall Filter Policers at the Same Interface

Example: Limiting Inbound Traffic at Your Network Border by Configuring an Ingress Single-Rate Two-Color Policer

IN THIS SECTION

- Requirements | **2044**
- Overview | **2044**
- Configuration | **2047**
- Verification | **2053**

This example shows you how to configure an ingress single-rate two-color policer to filter incoming traffic. The policer enforces the class-of-service (CoS) strategy for in-contract and out-of-contract traffic. You can apply a single-rate two-color policer to incoming packets, outgoing packets, or both. This example applies the policer as an input (ingress) policer. The goal of this topic is to provide you with an introduction to policing by using a example that shows traffic policing in action.

Policers use a concept known as a token bucket to allocate system resources based on the parameters defined for the policer. A thorough explanation of the token bucket concept and its underlying algorithms is beyond the scope of this document. For more information about traffic policing, and CoS in general, refer to *QOS-Enabled Networks—Tools and Foundations* by Miguel Barreiros and Peter Lundqvist. This book is available at many online booksellers and at www.juniper.net/books.

Requirements

To verify this procedure, this example uses a traffic generator. The traffic generator can be hardware-based or it can be software running on a server or host machine.

The functionality in this procedure is widely supported on devices that run Junos OS. The example shown here was tested and verified on MX Series routers running Junos OS Release 10.4.

Overview

IN THIS SECTION

- Topology | **2046**

Single-rate two-color policing enforces a configured rate of traffic flow for a particular service level by applying implicit or configured actions to traffic that does not conform to the limits. When you apply a single-rate two-color policer to the input or output traffic at an interface, the policer meters the traffic flow to the rate limit defined by the following components:

- Bandwidth limit—The average number of bits per second permitted for packets received or transmitted at the interface. You can specify the bandwidth limit as an absolute number of bits per second or as a percentage value from 1 through 100. If a percentage value is specified, the effective bandwidth limit is calculated as a percentage of either the physical interface media rate or the logical interface configured shaping rate.
- Burst-size limit—The maximum size permitted for bursts of data. Burst sizes are measured in bytes. We recommend two formulas for calculating burst size:

  Burst size = bandwidth x allowable time for burst traffic / 8

Or

Burst size = interface mtu x 10

For information about configuring the burst size, see "Determining Proper Burst Size for Traffic Policers" on page 1909.

> **NOTE**: There is a finite buffer space for an interface. In general, the estimated total buffer depth for an interface is about 125 ms.

For a traffic flow that conforms to the configured limits (categorized as green traffic), packets are implicitly marked with a packet loss priority (PLP) level of low and are allowed to pass through the interface unrestricted.

For a traffic flow that exceeds the configured limits (categorized as red traffic), packets are handled according to the traffic-policing actions configured for the policer. This example discards packets that burst over the 15 KBps limit.

To rate-limit Layer 3 traffic, you can apply a two-color policer in the following ways:

- Directly to a logical interface, at a specific protocol level.
- As the action of a standard stateless firewall filter that is applied to a logical interface, at a specific protocol level. This is the technique used in this example.

To rate-limit Layer 2 traffic, you can apply a two-color policer as a logical interface policer only. You cannot apply a two-color policer to Layer 2 traffic through a firewall filter.

> ⚠ **CAUTION**: You can choose either bandwidth-limit or bandwidth percent within the policer, as they are mutually exclusive. You cannot configure a policer to use bandwidth percent for aggregate, tunnel, and software interfaces.

In this example, the host is a traffic generator emulating a webserver. Devices R1 and R2 are owned by a service provider. The webserver is accessed by users on Device Host2. Device Host1 will be sending traffic with a source TCP HTTP port of 80 to the users. A single-rate two-color policer is configured and applied to the interface on Device R1 that connects to Device Host1. The policer enforces the contractual bandwidth availability made between the owner of the webserver and the service provider that owns Device R1 for the web traffic that flows over the link that connects Device Host1 to Device R1.

In accordance with the contractual bandwidth availability made between the owner of the webserver and the service provider that owns Devices R1 and R2, the policer will limit the HTTP port 80 traffic originating from Device Host1 to using 700 Mbps (70 percent) of the available bandwidth with an

allowable burst rate of 10 x the MTU size of the gigabit Ethernet interface between the host Device Host1 and Device R1.

NOTE: In a real-world scenario you would probably also rate limit traffic for a variety of other ports such as FTP, SFTP, SSH, TELNET, SMTP, IMAP, and POP3 because they are often included as additional services with web hosting services.

NOTE: You need to leave some additional bandwidth available that is not rate limited for network control protocols such as routing protocols, DNS, and any other protocols required to keep network connectivity operational. This is why the firewall filter has a final accept condition on it.

Topology

This example uses the topology in Figure 86 on page 2046.

Figure 86: Single-Rate Two-Color Policer Scenario

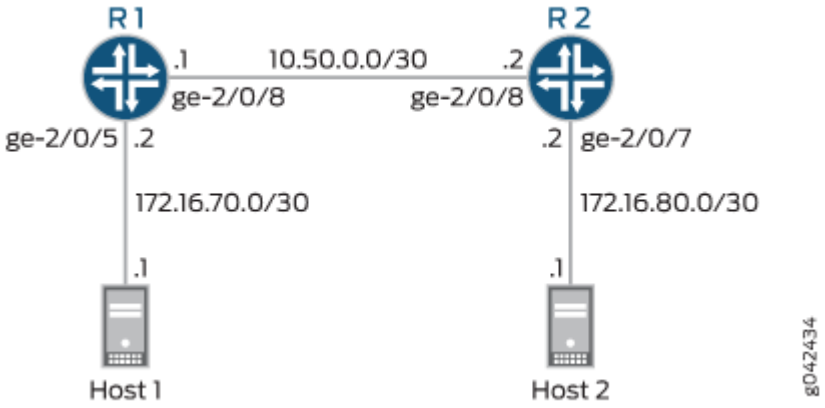
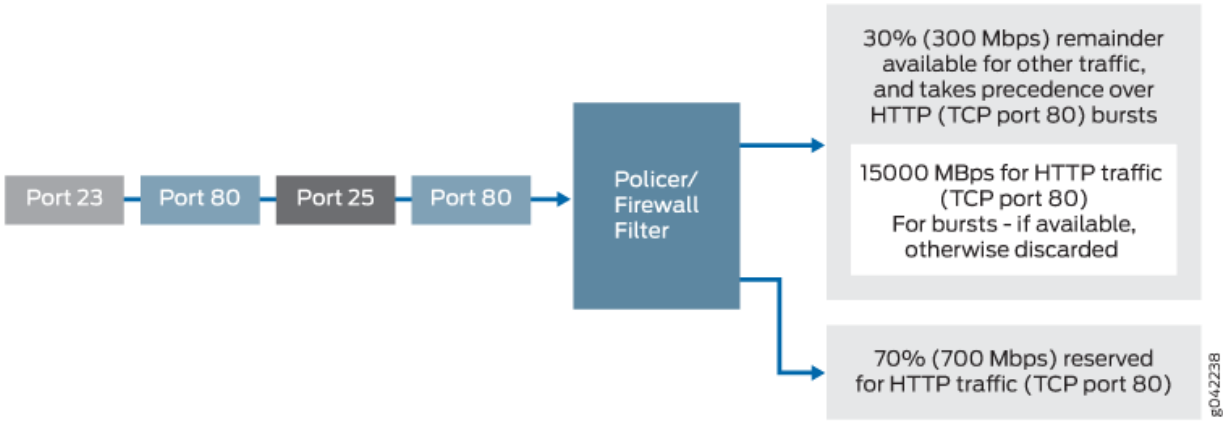


Figure 87 on page 2047 shows the policing behavior.

Figure 87: Traffic Limiting in a Single-Rate Two-Color Policer Scenario



Configuration

IN THIS SECTION

- Procedure | 2047

Procedure

CLI Quick Configuration

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

Device R1

```
set interfaces ge-2/0/5 description to-Host
set interfaces ge-2/0/5 unit 0 family inet address 172.16.70.2/30
set interfaces ge-2/0/5 unit 0 family inet filter input mf-classifier
set interfaces ge-2/0/8 description to-R2
set interfaces ge-2/0/8 unit 0 family inet address 10.50.0.1/30
set interfaces lo0 unit 0 description looback-interface
set interfaces lo0 unit 0 family inet address 192.168.13.1/32
set firewall policer discard if-exceeding bandwidth-limit 700m
```

```
set firewall policer discard if-exceeding burst-size-limit 15k
set firewall policer discard then discard
set firewall family inet filter mf-classifier term t1 from protocol tcp
set firewall family inet filter mf-classifier term t1 from port 80
set firewall family inet filter mf-classifier term t1 then policer discard
set firewall family inet filter mf-classifier term t2 then accept
set protocols ospf area 0.0.0.0 interface ge-2/0/5.0 passive
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/8.0
```

Device R2

```
set interfaces ge-2/0/8 description to-R1
set interfaces ge-2/0/8 unit 0 family inet address 10.50.0.2/30
set interfaces ge-2/0/7 description to-Host
set interfaces ge-2/0/7 unit 0 family inet address 172.16.80.2/30
set interfaces lo0 unit 0 description looback-interface
set interfaces lo0 unit 0 family inet address 192.168.14.1/32
set protocols ospf area 0.0.0.0 interface ge-2/0/7.0 passive
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/8.0
```

Step-by-Step Procedure

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

To configure Device R1:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set ge-2/0/5 description to-Host
user@R1# set ge-2/0/5 unit 0 family inet address 172.16.70.2/30
user@R1# set ge-2/0/8 description to-R2
user@R1# set ge-2/0/8 unit 0 family inet address 10.50.0.1/30
user@R1# set lo0 unit 0 description looback-interface
user@R1# set lo0 unit 0 family inet address 192.168.13.1/32
```

2. Apply the firewall filter to interface ge-2/0/5 as an input filter.

```
[edit interfaces ge-2/0/5 unit 0 family inet]
user@R1# set filter input mf-classifier
```

3. Configure the policer to rate-limit to a bandwidth of 700 Mbps and a burst size of 15000 KBps for HTTP traffic (TCP port 80).

```
[edit firewall policer discard]
user@R1# set if-exceeding bandwidth-limit 700m
user@R1# set if-exceeding burst-size-limit 15k
```

4. Configure the policer to discard packets in the red traffic flow.

```
[edit firewall policer discard]
user@R1# set then discard
```

5. Configure the two conditions of the firewall to accept all TCP traffic to port HTTP (port 80).

```
[edit firewall  family inet filter mf-classifier]
user@R1# set term t1 from protocol tcp
user@R1# set term t1 from port 80
```

6. Configure the firewall action to rate-limit HTTP TCP traffic using the policer.

```
[edit firewall  family inet filter mf-classifier]
user@R1# set term t1 then policer discard
```

7. At the end of the firewall filter, configure a default action that accepts all other traffic.

   Otherwise, all traffic that arrives on the interface and is not explicitly accepted by the firewall is discarded.

```
[edit firewall  family inet filter mf-classifier]
user@R1# set term t2 then accept
```

8. Configure OSPF.

```
[edit protocols ospf]
user@R1# set area 0.0.0.0 interface ge-2/0/5.0 passive
user@R1# set area 0.0.0.0 interface lo0.0 passive
user@R1# set area 0.0.0.0 interface ge-2/0/8.0
```

Step-by-Step Procedure

To configure Device R2:

1. Configure the device interfaces.

```
[edit interfaces]
user@R1# set ge-2/0/8 description to-R1
user@R1# set ge-2/0/7 description to-Host
user@R1# set lo0 unit 0 description looback-interface
user@R1# set ge-2/0/8 unit 0 family inet address 10.50.0.2/30
user@R1# set ge-2/0/7 unit 0 family inet address 172.16.80.2/30
user@R1# set lo0 unit 0 family inet address 192.168.14.1/32
```

2. Configure OSPF.

```
[edit protocols ospf]
user@R1# set area 0.0.0.0 interface ge-2/0/7.0 passive
user@R1# set area 0.0.0.0 interface lo0.0 passive
user@R1# set area 0.0.0.0 interface ge-2/0/8.0
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces` , `show firewall`, and `show protocols ospf` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R1# show interfaces
ge-2/0/5 {
    description to-Host;
    unit 0 {
```

```
        family inet {
            filter {
                input mf-classifier;
            }
            address 172.16.70.2/30;
        }
    }
}
ge-2/0/8 {
    description to-R2;
    unit 0 {
        family inet {
            address 10.50.0.1/30;
        }
    }
}
lo0 {
    unit 0 {
        description looback-interface;
        family inet {
            address 192.168.13.1/32;
        }
    }
}
```

```
user@R1# show firewall
family inet {
    filter mf-classifier {
        term t1 {
            from {
                protocol tcp;
                port 80;
            }
            then policer discard;
        }
        term t2 {
            then accept;
        }
    }
}
policer discard {
```

```
    if-exceeding {
        bandwidth-limit 700m;
        burst-size-limit 15k;
    }
    then discard;
}
```

```
user@R1# show protocols ospf
area 0.0.0.0 {
    interface ge-2/0/5.0 {
        passive;
    }
    interface lo0.0 {
        passive;
    }
    interface ge-2/0/8.0;
}
```

If you are done configuring Device R1, enter `commit` from configuration mode.

```
user@R2# show interfaces
ge-2/0/7 {
    description to-Host;
    unit 0 {
        family inet {
            address 172.16.80.2/30;
        }
    }
}
ge-2/0/8 {
    description to-R1;
    unit 0 {
        family inet {
            address 10.50.0.2/30;
        }
    }
}
lo0 {
    unit 0 {
        description looback-interface;
        family inet {
```

```
            address 192.168.14.1/32;
        }
    }
}
```

```
user@R2# show protocols ospf
area 0.0.0.0 {
    interface ge-2/0/7.0 {
        passive;
    }
    interface lo0.0 {
        passive;
    }
    interface ge-2/0/8.0;
}
```

If you are done configuring Device R2, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Clearing the Counters | **2053**
- Sending TCP Traffic into the Network and Monitoring the Discards | **2054**

Confirm that the configuration is working properly.

Clearing the Counters

Purpose

Confirm that the firewall counters are cleared.

Action

On Device R1, run the `clear firewall all` command to reset the firewall counters to 0.

```
user@R1> clear firewall all
```

Sending TCP Traffic into the Network and Monitoring the Discards

Purpose

Make sure that the traffic of interest that is sent is rate-limited on the input interface (ge-2/0/5).

Action

1. Use a traffic generator to send 10 TCP packets with a source port of 80.

   The -s flag sets the source port. The -k flag causes the source port to remain steady at 80 instead of incrementing. The -c flag sets the number of packets to 10. The -d flag sets the packet size.

   The destination IP address of 172.16.80.1 belongs to Device Host 2 that is connected to Device R2. The user on Device Host 2 has requested a webpage from Device Host 1 (the webserver emulated by the traffic generator on Device Host 1). The packets that being rate-limited are sent from Device Host 1 in response to the request from Device Host 2.

   NOTE: In this example the policer numbers are reduced to a bandwidth limit of 8 Kbps and a burst size limit of 1500 KBps to ensure that some packets are dropped during this test.

```
[root@host]#  hping 172.16.80.1  -c 10 -s 80  -k -d 300


[User@Host]#  hping 172.16.80.1 -c 10 -s 80 -k -d 350
HPING 172.16.80.1 (eth1 172.16.80.1): NO FLAGS are set, 40 headers + 350 data bytes
len=46 ip=172.16.80.1 ttl=62 DF id=0 sport=0 flags=RA seq=0 win=0 rtt=0.5 ms
.
.
.
--- 172.16.80.1 hping statistic ---
10 packets transmitted, 6 packets received, 40% packet loss
round-trip min/avg/max = 0.5/3000.8/7001.3 ms
```

2. On Device R1, check the firewall counters by using the `show firewall` command.

```
user@R1> show firewall


User@R1# run show firewall

Filter: __default_bpdu_filter__

Filter: mf-classifier
Policers:
Name                                                Bytes              Packets
discard-t1                                          1560               4
```

Meaning

In Steps 1 and 2 the output from both devices shows that 4 packets were discarded This means that there was at least 8 Kbps of green (in-contract HTTP port 80) traffic and that the 1500 KBps burst option for red out-of-contract HTTP port 80 traffic was exceeded.

Example: Configuring Interface and Firewall Filter Policers at the Same Interface

IN THIS SECTION

- Requirements | 2055
- Overview | 2056
- Configuration | 2057
- Verification | 2067

This example shows how to configure three single-rate two-color policers and apply the policers to the IPv4 input traffic at the same single-tag virtual LAN (VLAN) logical interface.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | **2056**

In this example, you configure three single-rate two-color policers and apply the policers to the IPv4 input traffic at the same single-tag VLAN logical interface. Two policers are applied to the interface through a firewall filter, and one policer is applied directly to the interface.

You configure one policer, named `p-all-1m-5k-discard`, to rate-limit traffic to 1 Mbps with a burst size of 5000 bytes. You apply this policer directly to IPv4 input traffic at the logical interface. When you apply a policer directly to protocol-specific traffic at a logical interface, the policer is said to be applied as an *interface policer*.

You configure the other two policers to allow burst sizes of 500 KB, and you apply these policers to IPv4 input traffic at the logical interface by using an IPv4 standard stateless firewall filter. When you apply a policer to protocol-specific traffic at a logical interface through a firewall filter action, the policer is said to be applied as a *firewall-filter policer*.

- You configure the policer named `p-icmp-500k-500k-discard` to rate-limit traffic to 500 Kbps with a burst size of 500 K bytes by discarding packets that do not conform to these limits. You configure one of the firewall filter terms to apply this policer to Internet Control Message Protocol (ICMP) packets.
- You configure the policer named `p-ftp-10p-500k-discard` to rate-limit traffic to a 10 percent bandwidth with a burst size of 500 KB by discarding packets that do not conform to these limits. You configure another firewall-filter term to apply this policer to File Transfer Protocol (FTP) packets.

A policer that you configure with a bandwidth limit expressed as a percentage value (rather than as an absolute bandwidth value) is called a *bandwidth policer*. Only single-rate two-color policers can be configured with a percentage bandwidth specification. By default, a bandwidth policer rate-limits traffic to the specified percentage of the line rate of the physical interface underlying the target logical interface.

Topology

You configure the target logical interface as a single-tag VLAN logical interface on a Fast Ethernet interface operating at 100 Mbps. This means that the policer you configure with the 10-percent bandwidth-limit (the policer that you apply to FTP packets) rate-limits the FTP traffic on this interface to 10 Mbps.

NOTE: In this example, you do not configure the bandwidth policer as a *logical-bandwidth policer*. Therefore, the percentage is based on the physical media rate rather than on the configured shaping rate of the logical interface.

The firewall filter that you configure to reference two of the policers must be configured as an *interface-specific filter*. Because the policer that is used to rate-limit FTP packets specifies the bandwidth limit as a percentage value, the firewall filter that references this policer must be configured as an interface-specific filter. Thus, if this firewall filter were to be applied to multiple interfaces instead of just the Fast Ethernet interface in this example, unique policers and counters would be created for each interface to which the filter is applied.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2057**
- Configuring the Single-Tag VLAN Logical Interface | **2058**
- Configuring the Three Policers | **2060**
- Configuring the IPv4 Firewall Filter | **2062**
- Applying the Interface Policer and Firewall Filter Policers to the Logical Interface | **2065**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces fe-0/1/1 vlan-tagging
set interfaces fe-0/1/1 unit 0 vlan-id 100
set interfaces fe-0/1/1 unit 0 family inet address 10.20.15.1/24
set interfaces fe-0/1/1 unit 1 vlan-id 101
```

```
set interfaces fe-0/1/1 unit 1 family inet address 10.20.240.1/24
set firewall policer p-all-1m-5k-discard if-exceeding bandwidth-limit 1m
set firewall policer p-all-1m-5k-discard if-exceeding burst-size-limit 5k
set firewall policer p-all-1m-5k-discard then discard
set firewall policer p-ftp-10p-500k-discard if-exceeding bandwidth-percent 10
set firewall policer p-ftp-10p-500k-discard if-exceeding burst-size-limit 500k
set firewall policer p-ftp-10p-500k-discard then discard
set firewall policer p-icmp-500k-500k-discard if-exceeding bandwidth-limit 500k
set firewall policer p-icmp-500k-500k-discard if-exceeding burst-size-limit 500k
set firewall policer p-icmp-500k-500k-discard then discard
set firewall family inet filter filter-ipv4-with-limits interface-specific
set firewall family inet filter filter-ipv4-with-limits term t-ftp from protocol tcp
set firewall family inet filter filter-ipv4-with-limits term t-ftp from port ftp
set firewall family inet filter filter-ipv4-with-limits term t-ftp from port ftp-data
set firewall family inet filter filter-ipv4-with-limits term t-ftp then policer p-ftp-10p-500k-
discard
set firewall family inet filter filter-ipv4-with-limits term t-icmp from protocol icmp
set firewall family inet filter filter-ipv4-with-limits term t-icmp then policer p-
icmp-500k-500k-discard
set firewall family inet filter filter-ipv4-with-limits term catch-all then accept
set interfaces fe-0/1/1 unit 1 family inet filter input filter-ipv4-with-limits
set interfaces fe-0/1/1 unit 1 family inet policer input p-all-1m-5k-discard
```

Configuring the Single-Tag VLAN Logical Interface

Step-by-Step Procedure

To configure the single-tag VLAN logical interface:

1. Enable configuration of the Fast Ethernet interface.

```
[edit]
user@host# edit interfaces fe-0/1/1
```

2. Enable single-tag VLAN framing.

```
[edit interfaces fe-0/1/1]
user@host# set vlan-tagging
```

3. Bind VLAN IDs to the logical interfaces.

```
[edit interfaces fe-0/1/1]
user@host# set unit 0 vlan-id 100
user@host# set unit 1 vlan-id 101
```

4. Configure IPv4 on the single-tag VLAN logical interfaces.

```
[edit interfaces fe-0/1/1]
user@host# set unit 0 family inet address 10.20.15.1/24
user@host# set unit 1 family inet address 10.20.240.1/24
```

Results

Confirm the configuration of the VLAN by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
fe-0/1/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.20.15.1/24;
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 10.20.240.1/24;
        }
    }
}
```

Configuring the Three Policers

Step-by-Step Procedure

To configure the three policers:

1. Enable configuration of a two-color policer that discards packets that do not conform to a bandwidth of 1 Mbps and a burst size of 5000 bytes.

   NOTE: You apply this policer directly to all IPv4 input traffic at the single-tag VLAN logical interface, so the packets will not be filtered before being subjected to rate limiting.

   ```
   [edit]
   user@host# edit firewall policer p-all-1m-5k-discard
   ```

2. Configure the first policer.

   ```
   [edit firewall policer p-all-1m-5k-discard]
   user@host# set if-exceeding bandwidth-limit 1m
   user@host# set if-exceeding burst-size-limit 5k
   user@host# set then discard
   ```

3. Enable configuration of a two-color policer that discards packets that do not conform to a bandwidth specified as “10 percent” and a burst size of 500,000 bytes.

   You apply this policer only to the FTP traffic at the single-tag VLAN logical interface.

   You apply this policer as the action of an IPv4 firewall filter term that matches FTP packets from TCP.

   ```
   [edit firewall policer p-all-1m-5k-discard]
   user@host# up

   [edit]
   user@host# edit firewall policer p-ftp-10p-500k-discard
   ```

4. Configure policing limits and actions.

```
[edit firewall policer p-ftp-10p-500k-discard]
user@host# set if-exceeding bandwidth-percent 10
user@host# set if-exceeding burst-size-limit 500k
user@host# set then discard
```

Because the bandwidth limit is specified as a percentage, the firewall filter that references this policer must be configured as an interface-specific filter.

> **NOTE**: If you wanted this policer to rate-limit to 10 percent of the logical interface configured shaping rate (rather than to 10 percent of the physical interface media rate), you would need to include the `logical-bandwidth-policer` statement at the `[edit firewall policer p-all-1m-5k-discard]` hierarchy level. This type of policer is called a *logical-bandwidth policer*.

5. Enable configuration of the IPv4 firewall filter policer for ICMP packets.

```
[edit firewall policer p-ftp-10p-500k-discard]
user@host# up

[edit]
user@host# edit firewall policer p-icmp-500k-500k-discard
```

6. Configure policing limits and actions.

```
[edit firewall policer p-icmp-500k-500k-discard]
user@host# set if-exceeding bandwidth-limit 500k
user@host# set if-exceeding burst-size-limit 500k
user@host# set then discard
```

Results

Confirm the configuration of the policers by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer p-all-1m-5k-discard {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 5k;
    }
    then discard;
}
policer p-ftp-10p-500k-discard {
    if-exceeding {
        bandwidth-percent 10;
        burst-size-limit 500k;
    }
    then discard;
}
policer p-icmp-500k-500k-discard {
    if-exceeding {
        bandwidth-limit 500k;
        burst-size-limit 500k;
    }
    then discard;
}
```

Configuring the IPv4 Firewall Filter

Step-by-Step Procedure

To configure the IPv4 firewall filter:

1. Enable configuration of the IPv4 firewall filter.

```
[edit]
user@host# edit firewall family inet filter filter-ipv4-with-limits
```

2. Configure the firewall filter as interface-specific.

```
[edit firewall family inet filter filter-ipv4-with-limits]
user@host# set interface-specific
```

The firewall filter must be interface-specific because one of the policers referenced is configured with a bandwidth limit expressed as a percentage value.

3. Enable configuration of a filter term to rate-limit FTP packets.

```
[edit firewall family inet filter filter-ipv4-with-limits]
user@host# edit term t-ftp

[edit firewall family inet filter filter-ipv4-with-limits term t-ftp]
user@host# set from protocol tcp
user@host# set from port [ ftp ftp-data ]
```

FTP messages are sent over TCP port 20 (`ftp`) and received over TCP port 21 (`ftp-data`).

4. Configure the filter term to match FTP packets.

```
[edit firewall family inet filter filter-ipv4-with-limits term t-ftp]
user@host# set then policer p-ftp-10p-500k-discard
```

5. Enable configuration of a filter term to rate-limit ICMP packets.

```
[edit firewall family inet filter filter-ipv4-with-limits term t-ftp]
user@host# up

[edit firewall family inet filter filter-ipv4-with-limits]
user@host# edit term t-icmp
```

6. Configure the filter term for ICMP packets

```
[edit firewall family inet filter filter-ipv4-with-limits term t-icmp]
user@host# set from protocol icmp
user@host# set then policer p-icmp-500k-500k-discard
```

7. Configure a filter term to accept all other packets without policing.

```
[edit firewall family inet filter filter-ipv4-with-limits term t-icmp]
user@host# up

[edit firewall family inet filter filter-ipv4-with-limits]
user@host# set term catch-all then accept
```

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter-ipv4-with-limits {
        interface-specific;
        term t-ftp {
            from {
                protocol tcp;
                port [ ftp ftp-data ];
            }
            then policer p-ftp-10p-500k-discard;
        }
        term t-icmp {
            from {
                protocol icmp;
            }
            then policer p-icmp-500k-500k-discard;
        }
        term catch-all {
            then accept;
        }
    }
}
policer p-all-1m-5k-discard {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 5k;
```

```
    }
    then discard;
}
policer p-ftp-10p-500k-discard {
    if-exceeding {
        bandwidth-percent 10;
        burst-size-limit 500k;
    }
    then discard;
}
policer p-icmp-500k-500k-discard {
    if-exceeding {
        bandwidth-limit 500k;
        burst-size-limit 500k;
    }
    then discard;
}
```

Applying the Interface Policer and Firewall Filter Policers to the Logical Interface

Step-by-Step Procedure

To apply the three policers to the VLAN:

1. Enable configuration of IPv4 on the logical interface.

```
[edit]
user@host# edit interfaces fe-0/1/1 unit 1 family inet
```

2. Apply the firewall filter policers to the interface.

```
[edit interfaces fe-0/1/1 unit 1 family inet]
user@host# set filter input filter-ipv4-with-limits
```

3. Apply the interface policer to the interface.

```
[edit interfaces fe-0/1/1 unit 1 family inet]
user@host# set policer input p-all-1m-5k-discard
```

Input packets at `fe-0/1/1.0` are evaluated against the interface policer before they are evaluated against the firewall filter policers. For more information, see "Order of Policer and Firewall Filter Operations" on page 1926.

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
fe-0/1/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.20.15.1/24;
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            filter {
                input filter-ipv4-with-limits;
            }
            policer {
                input p-all-1m-5k-discard;
            }
            address 10.20.240.1/24;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Policers Applied Directly to the Logical Interface | 2067
- Displaying Statistics for the Policer Applied Directly to the Logical Interface | 2067
- Displaying the Policers and Firewall Filters Applied to an Interface | 2068
- Displaying Statistics for the Firewall Filter Policers | 2069

Confirm that the configuration is working properly.

Displaying Policers Applied Directly to the Logical Interface

Purpose

Verify that the interface policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces policers` operational mode command for logical interface `fe-0/1/1.1`. The command output section for the **Proto** column and **Input Policer** column shows that the policer `p-all-1m-5k-discard` is evaluated when packets are received on the logical interface.

```
user@host> show interfaces policers fe-0/1/1.1
Interface       Admin Link Proto Input Policer         Output Policer
fe-0/1/1.1      up    up
                           inet  p-all-1m-5k-discard-fe-0/1/1.1-inet-i
```

In this example, the interface policer is applied to logical interface traffic in the input direction only.

Displaying Statistics for the Policer Applied Directly to the Logical Interface

Purpose

Verify the number of packets evaluated by the interface policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction.

```
user@host> show policer p-all-1m-5k-discard-fe-0/1/1.1-inet-i
Policers:
Name                                                Bytes              Packets
p-all-1m-5k-discard-fe-0/1/1.1-inet-i                 200                    5
```

Displaying the Policers and Firewall Filters Applied to an Interface

Purpose

Verify that the firewall filter `filter-ipv4-with-limits` is applied to the IPv4 input traffic at logical interface `fe-0/1/1.1`.

Action

Use the `show interfaces statistics` operational mode command for logical interface `fe-0/1/1.1`, and include the `detail` option. Under the **Protocol inet** section of the command output section, the **Input Filters** and **Policer** lines display the names of filter and policer applied to the logical interface in the input direction.

```
user@host> show interfaces statistics fe-0/1/1.1 detail
  Logical interface fe-0/1/1.1 (Index 83) (SNMP ifIndex 545) (Generation 153)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.100 ]  Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Transit statistics:
     Input  bytes  :                   0                   0 bps
```

```
     Output bytes  :                    0                    0 bps
     Input  packets:                    0                    0 pps
     Output packets:                    0                    0 pps
    Protocol inet, MTU: 1500, Generation: 176, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter-ipv4-with-limits-fe-0/1/1.1-i
      Policer: Input: p-all-1m-5k-discard-fe-0/1/1.1-inet-i
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.20.130/24, Local: 10.20.130.1, Broadcast: 10.20.130.255,
        Generation: 169
```

In this example, the two firewall filter policers are applied to logical interface traffic in the input direction only.

Displaying Statistics for the Firewall Filter Policers

Purpose

Verify the number of packets evaluated by the firewall filter policers.

Action

Use the `show firewall` operational mode command for the filter you applied to the logical interface.

```
[edit]
user@host> show firewall filter filter-ipv4-with-limits-fe-0/1/1.1-i

Filter: filter-ipv4-with-limits-fe-0/1/1.1-i
Policers:
Name                                                Bytes              Packets
p-ftp-10p-500k-discard-t-ftp-fe-0/1/1.1-i               0                    0
p-icmp-500k-500k-discard-t-icmp-fe-0/1/1.1-i            0                    0
```

The command output displays the names of the policers (`p-ftp-10p-500k-discard` and `p-icmp-500k-500k-discard`), combined with the names of the filter terms (`t-ftp` and `t-icmp`, respectively) under which the policer action is specified. The policer-specific output lines display the number of packets that matched the filter term. This is only the number of out-of-specification (out-of-spec) packet counts, not all packets policed by the policer.

SEE ALSO

- Order of Policer and Firewall Filter Operations | 1926
- Two-Color Policer Configuration Overview | 2034
- Single-Rate Two-Color Policer Overview
- Example: Limiting Inbound Traffic at Your Network Border by Configuring an Ingress Single-Rate Two-Color Policer

RELATED DOCUMENTATION

- Order of Policer and Firewall Filter Operations | 1926
- Two-Color Policer Configuration Overview | 2034
- Guidelines for Applying Traffic Policers | 1934
- Determining Proper Burst Size for Traffic Policers | 1909

Bandwidth Policers

IN THIS SECTION

- Bandwidth Policer Overview | 2070
- Example: Configuring a Logical Bandwidth Policer | 2072

Bandwidth Policer Overview

IN THIS SECTION

- Guidelines for Configuring a Bandwidth Policer | 2071
- Guidelines for Applying a Bandwidth Policer | 2071

For a single-rate two-color policer only, you can specify the bandwidth limit as a percentage value from 1 through 100 instead of as an absolute number of bits per second. This type of two-color policer, called

a *bandwidth policer*, rate-limits traffic to a bandwidth limit that is calculated as a percentage of either the physical interface media rate or the *logical interface* configured shaping rate.

Guidelines for Configuring a Bandwidth Policer

The following guidelines apply to configuring a bandwidth policer:

- To specify a percentage bandwidth limit, you include the `bandwidth-percent` *percentage* statement in place of the `bandwidth-limit` *bps* statement.
- By default, a bandwidth policer calculates the percentage bandwidth limit based on the physical interface port speed. To configure a bandwidth policer to calculate the percentage bandwidth limit based on the configured logical interface shaping rate instead, include the `logical-bandwidth-policer` statement at the `[edit firewall policer` *policer-name*`]` hierarchy level. This type of bandwidth policer is called a *logical bandwidth policer*.

  You can configure a logical interface shaping rate by including the `shaping-rate` *bps* statement at the `[edit class-of-service interfaces interface` *interface-name* `unit` *logical-unit-number*`]` hierarchy level. A logical interface shaping rate causes the specified amount of bandwidth to be allocated to the logical interface.

  > **NOTE**: If you configure a logical-bandwidth policer and then apply the policer to a logical interface that is not configured with a shaping rate, then the policer rate-limits traffic on that logical interface to calculate the percentage bandwidth limit based on the physical interface port speed, even if you include the `logical-bandwidth-policer` statement in the bandwidth policer configuration.

- If you reference a bandwidth policer from a stateless *firewall filter* term, you must include the `interface-specific` statement in the firewall filter configuration.

Guidelines for Applying a Bandwidth Policer

The following guidelines pertain to applying a bandwidth policer to traffic:

- You can use a bandwidth policer to rate-limit protocol-specific traffic (not `family any`) at the input or output of a logical interface.
- You can apply a bandwidth policer directly to protocol-specific input or output traffic at a logical interface.
- To send only selected packets to a bandwidth policer, you can reference the bandwidth policer from a stateless firewall filter term and then apply the filter to logical interface traffic for a specific protocol family.

  - To reference a *logical bandwidth policer* from a firewall filter, you must include the `interface-specific` statement in the firewall filter configuration.
  - You cannot use a bandwidth policer for forwarding-table filters.
- You cannot apply a bandwidth policer to an aggregate interface, a tunnel interface, or a software interface.

SEE ALSO

> Two-Color Policer Configuration Overview | **2034**
>
> Example: Configuring a Logical Bandwidth Policer
>
> *bandwidth-percent*
>
> *interface-specific*
>
> *logical-bandwidth-policer*
>
> *shaping-rate (Applying to an Interface)*

Example: Configuring a Logical Bandwidth Policer

IN THIS SECTION

- Requirements | **2072**
- Overview | **2073**
- Configuration | **2074**
- Verification | **2080**

This example shows how to configure a logical bandwidth policer.

Requirements

Before you begin, make sure that you have two logical units available on a Gigabit Ethernet interface.

Overview

IN THIS SECTION

- Topology | 2073

In this example, you configure a single-rate two-color policer that specifies the bandwidth limit as a percentage value rather than as an absolute number of bits per second. This type of policer is called a *bandwidth policer*. By default, a bandwidth policer enforces a bandwidth limit based on the line rate of the underlying physical interface. As an option, you can configure a bandwidth policer to enforce a bandwidth limit based on the configured shaping rate of the logical interface. To configure this type of bandwidth policer, called a *logical bandwidth policer*, you include the `logical-bandwidth-policer` statement in the policer configuration.

To configure a logical interface shaping rate, include the `shaping-rate` *`bps`* statement at the `[edit class-of-service interfaces interface` *`interface-name`* `unit` *`logical-unit-number`*`]` hierarchy level. This class-of-service (CoS) configuration statement causes the specified amount of bandwidth to be allocated to the logical interface.

NOTE: If you configure a policer bandwidth limit as a percentage but a shaping rate is not configured for the target logical interface, the policer bandwidth limit is calculated as a percentage of the physical interface media rate, even if you enable the logical-bandwidth policing feature.

To apply a logical bandwidth policer to a logical interface, you can apply the policer directly to the logical interface at the protocol family level or (if you only need to rate-limit filtered packets) you can reference the policer from a stateless firewall filter configured to operate in *interface-specific* mode.

Topology

In this example, you configure two logical interfaces on a single Gigabit Ethernet interface and configure a shaping rate on each logical interface. On logical interface `ge-1/3/0.0`, you allocate 4 Mbps of bandwidth. On logical interface `ge-1/3/0.1`, you allocate 2 Mbps of bandwidth.

You also configure a logical bandwidth policer with a bandwidth limit of 50 percent and a maximum burst size of 125,000 bytes, and then you apply the policer to input and output traffic at the logical units configured on `ge-1/3/0.0`. For logical interface `ge-1/3/0.0`, the policer rate-limits to a bandwidth limit of 2 Mbps (50 percent of the 4 Mbps shaping rate configured for the logical interface). For logical interface

ge-1/3/0.1, the policer rate-limits traffic to a bandwidth limit of 1 Mbps (50 percent of the 2 Mbps shaping rate configured for the logical interface).

If no shaping rate is configured for a target logical interface, the policer rate-limits to a bandwidth limit calculated as 50 percent of the physical interface media rate. For example, if you apply a 50 percent bandwidth policer to input or output traffic at a Gigabit Ethernet logical interface without rate shaping, the policer applies a bandwidth limit of 500 Mbps (50 percent of 1000 Mbps).

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2074**
- Configuring the Logical Interfaces | **2075**
- Configuring Traffic Rate-Shaping by Specifying the Amount of Bandwidth to be Allocated to the Logical Interface | **2076**
- Configuring the Logical Bandwidth Policer | **2077**
- Applying the Logical Bandwidth Policers to the Logical Interfaces | **2078**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the [edit] hierarchy level.

```
set interfaces ge-1/3/0 per-unit-scheduler
set interfaces ge-1/3/0 vlan-tagging
set interfaces ge-1/3/0 unit 0 vlan-id 100
set interfaces ge-1/3/0 unit 0 family inet address 172.16.1.1/30
set interfaces ge-1/3/0 unit 1 vlan-id 200
set interfaces ge-1/3/0 unit 1 family inet address 172.16.2.1/30
set class-of-service interfaces ge-1/3/0 unit 0 shaping-rate 4m
set class-of-service interfaces ge-1/3/0 unit 1 shaping-rate 2m
set firewall policer LB-policer logical-bandwidth-policer
```

```
set firewall policer LB-policer if-exceeding bandwidth-percent 50
set firewall policer LB-policer if-exceeding burst-size-limit 125k
set firewall policer LB-policer then discard
set interfaces ge-1/3/0 unit 0 family inet policer input LB-policer
set interfaces ge-1/3/0 unit 0 family inet policer output LB-policer
set interfaces ge-1/3/0 unit 1 family inet policer input LB-policer
set interfaces ge-1/3/0 unit 1 family inet policer output LB-policer
```

Configuring the Logical Interfaces

Step-by-Step Procedure

To configure the logical interfaces:

1. Enable configuration of the physical interface.

```
[edit]
user@host# edit interfaces ge-1/3/0

[edit interfaces ge-1/3/0]
user@host# set per-unit-scheduler
user@host# set vlan-tagging
```

2. Configure the first logical interface.

```
[edit interfaces ge-1/3/0]
user@host# set  unit 0 vlan-id 100
user@host# set  unit 0 family inet address 172.16.1.1/30
```

3. Configure the second logical interface.

```
[edit interfaces ge-1/3/0]
user@host# set unit 1 vlan-id 200
user@host# set unit 1 family inet address 172.16.2.1/30
```

Results

Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
 ge-1/3/0 {
    per-unit-scheduler;
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 172.16.1.1/30;
        }
    }
    unit 1 {
        vlan-id 200;
        family inet {
            address 172.16.2.1/30;
        }
    }
}
```

Configuring Traffic Rate-Shaping by Specifying the Amount of Bandwidth to be Allocated to the Logical Interface

Step-by-Step Procedure

To configure rate shaping by specifying the bandwidth to be allocated to the logical interface:

1. Enable CoS configuration on the physical interface.

```
[edit]
user@host# edit class-of-service interfaces ge-1/3/0
```

2. Configure rate shaping for the logical interfaces.

```
[edit class-of-service interfaces ge-1/3/0]
user@host# set unit 0 shaping-rate 4m
user@host# set unit 1 shaping-rate 2m
```

These statements allocate 4 Mbps of bandwidth to logical unit `ge-1/3/0.0` and 2 Mbps of bandwidth to logical unit `ge-1/3/0.1`.

Results

Confirm the configuration of the rate shaping by entering the `show class-of-service` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show class-of-service
interfaces {
    ge-1/3/0 {
        unit 0 {
            shaping-rate 4m;
        }
        unit 1 {
            shaping-rate 2m;
        }
    }
}
```

Configuring the Logical Bandwidth Policer

Step-by-Step Procedure

To configure the logical bandwidth policer:

1. Enable configuration of a single-rate two-color policer.

```
[edit]
user@host# edit firewall policer LB-policer
```

2. Configure the policer as a logical-bandwidth policer.

```
[edit firewall policer LB-policer]
user@host# set logical-bandwidth-policer
```

This applies the rate-limiting to logical interfaces.

3. Configure the policer traffic limits and actions.

```
[edit firewall policer LB-policer]
user@host# set if-exceeding bandwidth-percent 50
user@host# set if-exceeding burst-size-limit 125k
user@host# set then discard
```

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer LB-policer {
    logical-bandwidth-policer;
    if-exceeding {
        bandwidth-percent 50;
        burst-size-limit 125k;
    }
    then discard;
}
```

Applying the Logical Bandwidth Policers to the Logical Interfaces

Step-by-Step Procedure

To configure the logical bandwidth policers to the logical interfaces:

1. Enable configuration of the interface.

```
[edit]
user@host# edit interfaces ge-1/3/0
```

2. Apply the logical bandwidth policer to the first logical interface.

```
[edit interfaces ge-1/3/0]
user@host# set unit 0 family inet policer input LB-policer
user@host# set unit 0 family inet policer output LB-policer
```

3. Apply the policing to the second logical interface.

```
[edit interfaces ge-1/3/0]
user@host# set unit 1 family inet policer input LB-policer
user@host# set unit 1 family inet policer output LB-policer
```

Results

Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/0 {
    per-unit-scheduler;
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            policer {
                input LB-policer;
                output LB-policer;
            }
            address 172.16.1.1/30;
        }
    }
    unit 1 {
```

```
        vlan-id 200;
        family inet {
            policer {
                input LB-policer;
                output LB-policer;
            }
            address 172.16.2.1/30;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | **2080**
- Displaying Statistics for the Policer | **2082**

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interfaces `ge-1/3/0.0` and `ge-1/3/0.1`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface, and the **Protocol inet** section contains a **Policer** field that lists the policer `LB-policer` as an input or output policer as follows:

- **Input: LB-policer-ge-1/3/0.0-inet-i**
- **Output: LB-policer-ge-1/3/0.0-inet-o**

In this example, the policer is applied to logical interface traffic in both the input and output directions.

```
user@host> show interfaces ge-1/3/0.0 detail
  Logical interface ge-1/3/0.0 (Index 80) (SNMP ifIndex 154) (Generation 150)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.100 ]  Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Transit statistics:
     Input  bytes  :                   0                  0 bps
     Output bytes  :                   0                  0 bps
     Input  packets:                   0                  0 pps
     Output packets:                   0                  0 pps
    Protocol inet, MTU: 1500, Generation: 174, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Policer: Input: LB-policer-ge-1/3/0.0-inet-i, Output: LB-policer-ge-1/3/0.0-inet-o
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 172.16.1.0/30, Local: 172.16.1.1, Broadcast: 172.16.1.3, Generation: 165


user@host> show interfaces ge-1/3/0.1 detail
  Logical interface ge-1/3/0.1 (Index 81) (SNMP ifIndex 543) (Generation 151)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.200 ]  Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                  46
     Input  packets:                   0
     Output packets:                   1
    Transit statistics:
     Input  bytes  :                   0                  0 bps
```

```
     Output bytes  :                    0                    0 bps
     Input  packets:                    0                    0 pps
     Output packets:                    0                    0 pps
   Protocol inet, MTU: 1500, Generation: 175, Route table: 0
     Flags: Sendbcast-pkt-to-re
     Policer: Input: LB-policer-ge-1/3/0.1-inet-i, Output: LB-policer-ge-1/3/0.1-inet-o
     Addresses, Flags: Is-Preferred Is-Primary
       Destination: 172.17.1.0/30, Local: 172.17.1.1, Broadcast: 172.17.1.3, Generation: 167
```

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction. For the policer `LB-policer`, the input and output policer names are displayed as follows:

- **LB-policer-ge-1/3/0.0-inet-i**
- **LB-policer-ge-1/3/0.0-inet-o**
- **LB-policer-ge-1/3/0.1-inet-i**
- **LB-policer-ge-1/3/0.1-inet-o**

The **-inet-i** suffix denotes a policer applied to logical interface input traffic, while the **-inet-o** suffix denotes a policer applied to logical interface output traffic. In this example, the policer is applied to both input and output traffic on logical interface `ge-1/3/0.0` and logical interface `ge-1/3/0.1`.

```
user@host> show policer
Policers:
Name                                                Packets
__default_arp_policer__                                   0
LB-policer-ge-1/3/0.0-inet-i                              0
LB-policer-ge-1/3/0.0-inet-o                              0
LB-policer-ge-1/3/0.1-inet-i                              0
LB-policer-ge-1/3/0.1-inet-o                              0
```

SEE ALSO

- Two-Color Policer Configuration Overview | 2034
- Bandwidth Policer Overview
- *bandwidth-percent*
- *interface-specific*
- *logical-bandwidth-policer*
- *shaping-rate (Applying to an Interface)*

RELATED DOCUMENTATION

- Two-Color Policer Configuration Overview | 2034
- Guidelines for Applying Traffic Policers | 1934
- *bandwidth-percent*
- *interface-specific (Firewall Filters)*
- *logical-bandwidth-policer*
- *shaping-rate (Applying to an Interface)*

Prefix-Specific Counting and Policing Actions

IN THIS SECTION

- Prefix-Specific Counting and Policing Overview | 2084
- Filter-Specific Counter and Policer Set Overview | 2087
- Filter-Specific Policer Overview | 2087
- Example: Configuring Prefix-Specific Counting and Policing | 2088
- Prefix-Specific Counting and Policing Configuration Scenarios | 2098

Prefix-Specific Counting and Policing Overview

IN THIS SECTION

- Separate Counting and Policing for Each IPv4 Address Range | **2084**
- Prefix-Specific Action Configuration | **2084**
- Counter and Policer Set Size and Indexing | **2085**

Separate Counting and Policing for Each IPv4 Address Range

Prefix-specific counting and policing enables you to configure an IPv4 *firewall filter* term that matches on a source or destination address, applies a single-rate two-color policer as the term action, but associates the matched packet with a specific counter and policer instance based on the source or destination in the packet header. You can implicitly create a separate counter or policer instance for a single address or for a group of addresses.

Prefix-specific counting and policing uses a *prefix-specific action* configuration that specifies the name of the policer you want to apply, whether prefix-specific counting is to be enabled, and a source or destination address prefix range.

The prefix range specifies between 1 and 16 sequential set bits of an IPv4 address mask. The length of the prefix range determines the size of the counter and policer *set*, which consists of as few as 2 or as many as 65,536 counter and policer instances. The position of the bits of the prefix range determines the indexing of filter-matched packets into the set of instances.

NOTE: A prefix-specific action is specific to a source or destination *prefix range*, but it is not specific to a particular source or destination *address range*, and it is not specific to a particular interface.

To apply a prefix-specific action to the traffic at an interface, you configure a firewall filter term that matches on source or destination addresses, and then you apply the firewall filter to the interface. The flow of filtered traffic is rate-limited using prefix-specific counter and policer instances that are selected per packet based on the source or destination address in the header of the filtered packet.

Prefix-Specific Action Configuration

To configure a prefix-specific action, you specify the following information:

- Prefix-specific action name—Name that can be referenced as the action of an IPv4 standard firewall filter term that matches packets on source or destination addresses.
- Policer name—Name of a single-rate two-color policer for which you want to implicitly create prefix-specific instances.

  NOTE: For aggregated Ethernet interfaces, you can configure a prefix-specific action that references a logical interface policer (also called an aggregate policer). You can reference this type of prefix-specific action from an IPv4 standard firewall filter and then apply the filter at the aggregate level of the interface.

- Counting option—Option to include if you want to enable prefix-specific counters.
- Filter-specific option—Option to include if you want a single counter and policer set to be shared across all terms in the firewall filter. A prefix-specific action that operates in this way is said to operate in *filter-specific* mode. If you do not enable this option, the prefix-specific action operates in *term-specific* mode, meaning that a separate counter and policer set is created for each filter term that references the prefix-specific action.
- Source address prefix length—Length of the address prefix, from 0 through 32, to be used with a packet matched on the source address.
- Destination address prefix length—Length of the address prefix, from 0 through 32, to be used with a packet matched on the destination address.
- Subnet prefix length—Length of the subnet prefix, from 0 through 32, to be used with a packet matched on either the source or destination address.

You must configure source and destination address prefix lengths to be from 1 to 16 bits longer than the subnet prefix length. If you configure source or destination address prefix lengths to be more than 16 bits beyond the configured subnet prefix length, an error occurs when you try to commit the configuration.

Counter and Policer Set Size and Indexing

The number of prefix-specific actions (counters or policers) implicitly created for a prefix-specific action is determined by the length of the address prefix and the length of the subnet prefix:

1. Size of Counter and Policer Set = 2^(*source-or-destination-prefix-length* - *subnet-prefix-length*)

Table 116 on page 2086 shows examples of counter and policer set size and indexing.

Table 116: Examples of Counter and Policer Set Size and Indexing

Example Prefix Lengths Specified in the Prefix-Specific Action	Calculation of Counter or Policer Set Size	Indexing of Instances	
source-prefix-length = 32 *subnet-prefix-length* = 16	Size = 2^(32 - 16) = 2^16 = 65,536 instances **NOTE**: This calculation shows the largest counter or policer set size supported.	Instance 0:	*x.x*.0.0
		Instance 1:	*x.x*.0.1
		Instance 65535:	*x.x*.255.255
source-prefix-length = 32 *subnet-prefix-length* = 24	Size = 2^(32 - 24) = 2^8 = 256 instances	Instance 0:	*x.x.x*.0
		Instance 1:	*x.x.x*.1
		Instance 255:	*x.x.x*.255
source-prefix-length = 32 *subnet-prefix-length* = 25	Size = 2^(32 - 25) = 2^7 = 128 instances	Instance 0:	*x.x.x*.0
		Instance 1:	*x.x.x*.1
		Instance 127:	*x.x.x*.127
source-prefix-length = 24 *subnet-prefix-length* = 20	Size = 2^(24 - 20) = 2^4 = 16 instances	Instance 0:	*x.x*.0.*x*
		Instance 1:	*x.x*.1.*x*
		Instance 15:	*x.x*.15.*x*

SEE ALSO

Two-Color Policer Configuration Overview | **2034**

Filter-Specific Counter and Policer Set Overview

Example: Configuring Prefix-Specific Counting and Policing

Prefix-Specific Counting and Policing Configuration Scenarios

prefix-action (Configuring)

prefix-action (Firewall Filter Action)

Filter-Specific Counter and Policer Set Overview

By default, a prefix-specific policer set operates in *term-specific* mode so that, for a given *firewall filter*, the Junos OS creates a separate counter and policer set for every filter term that references the prefix-specific action. As an option, you can configure a prefix-specific policer set to operate in *filter-specific* mode so that a single prefix-specific policer set is used by all terms (within the same firewall filter) that reference the policer.

For an IPv4 firewall filter with multiple terms that reference the same prefix-specific policer set, configuring the policer set to operate in filter-specific mode enables you to count and monitor the activity of the policer set at the firewall filter level.

NOTE: Term-specific mode and filter-specific mode also apply to policers. See Filter-Specific Policer Overview.

To enable a prefix-specific policer set to operate in filter-specific mode, you can include the `filter-specific` statement at the following the hierarchy levels:

- `[edit firewall family inet prefix-action` *`prefix-action-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `firewall family inet prefix-action` *`prefix-action-name`*`]`

You can reference filter-specific, prefix-specific policer sets from IPv4 (`family inet`) firewall filters only.

SEE ALSO

Two-Color Policer Configuration Overview | **2034**

Prefix-Specific Counting and Policing Overview

Example: Configuring Prefix-Specific Counting and Policing

Prefix-Specific Counting and Policing Configuration Scenarios

Filter-Specific Policer Overview

By default, a policer operates in *term-specific* mode so that, for a given *firewall filter*, the Junos OS creates a separate policer instance for every filter term that references the policer. As an option, you can configure a policer to operate in *filter-specific* mode so that a single policer instance is used by all terms (within the same firewall filter) that reference the policer.

For an IPv4 firewall filter with multiple terms that reference the same policer, configuring the policer to operate in filter-specific mode enables you to count and monitor the activity of the policer at the firewall filter level.

NOTE: Term-specific mode and filter-specific mode also apply to prefix-specific policer sets.

To enable a single-rate two-color policer to operate in filter-specific mode, you can include the `filter-specific` statement at the following hierarchy levels:

- `[edit firewall policer policer-name]`
- `[edit logical-systems logical-system-name firewall policer policer-name]`

You can reference filter-specific policers from IPv4 (`family inet`) firewall filters only.

Example: Configuring Prefix-Specific Counting and Policing

IN THIS SECTION

- Requirements | **2088**
- Overview | **2088**
- Configuration | **2090**
- Verification | **2096**

This example shows how to configure prefix-specific counting and policing.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | **2089**

In this example, you configure prefix-specific counting and policing based on the last octet of the source address field in packets matched by an IPv4 firewall filter.

The single-rate two-color policer named `1Mbps-policer` rate-limits traffic to a bandwidth of 1,000,000 bps and a burst-size limit of 63,000 bytes, discarding any packets in a traffic flow that exceeds the traffic limits.

Independent of the IPv4 addresses contained in any packets passed from a firewall filter, the prefix-specific action named `psa-1Mbps-per-source-24-32-256` specifies a set of 256 counters and policers, numbered from 0 through 255. For each packet, the last octet of the source address field is used to index into the associated prefix-specific counter and policer in the set:

- Packets with a source address ending with the octet 0x0000 00000 index the first counter and policer in the set.
- Packets with a source address ending with the octet 0x0000 0001 index the second counter and policer in the set.
- Packets with a source address ending with the octet 0x1111 1111 index the last counter and policer in the set.

The `limit-source-one-24` firewall filter contains a single term that matches all packets from the `/24` subnet of source address `10.10.10.0`, passing these packets to the prefix-specific action `psa-1Mbps-per-source-24-32-256`.

Topology

In this example, because the filter term matches the `/24` subnet of a single source address, each counting and policing instance in the prefix-specific set is used for only one source address.

- Packets with a source address `10.10.10.0` index the first counter and policer in the set.
- Packets with a source address `10.10.10.1` index the second counter and policer in the set.
- Packets with a source address `10.10.10.255` index the last counter and policer in the set.

This example shows the simplest case of prefix-specific actions, in which the filter term matches on one address with a prefix length that is the same as the prefix length specified in the prefix-specific action for indexing into the set of prefix-specific counters and policers.

For descriptions of other configurations for prefix-specific counting and policing, see Prefix-Specific Counting and Policing Configuration Scenarios.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 2090
- Configuring a Policer for Prefix-Specific Counting and Policing | 2091
- Configuring a Prefix-Specific Action Based on the Policer | 2092
- Configuring an IPv4 Filter That References the Prefix-Specific Action | 2093
- Applying the Firewall Filter to IPv4 Input Traffic at a Logical Interface | 2095

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall policer 1Mbps-policer if-exceeding bandwidth-limit 1m
set firewall policer 1Mbps-policer if-exceeding burst-size-limit 63k
set firewall policer 1Mbps-policer then discard
set firewall family inet prefix-action psa-1Mbps-per-source-24-32-256 policer 1Mbps-policer
set firewall family inet prefix-action psa-1Mbps-per-source-24-32-256 count
set firewall family inet prefix-action psa-1Mbps-per-source-24-32-256 subnet-prefix-length 24
set firewall family inet prefix-action psa-1Mbps-per-source-24-32-256 source-prefix-length 32
set firewall family inet filter limit-source-one-24 term one from source-address 10.10.10.0/24
set firewall family inet filter limit-source-one-24 term one then prefix-action psa-1Mbps-per-
source-24-32-256
set interfaces so-0/0/2 unit 0 family inet filter input limit-source-one-24
set interfaces so-0/0/2 unit 0 family inet address 10.39.1.1/16
```

Configuring a Policer for Prefix-Specific Counting and Policing

Step-by-Step Procedure

To configure a policer to be used for prefix-specific counting and policing:

1. Enable configuration of a single-rate two-color policer.

```
[edit]
user@host# edit firewall policer 1Mbps-policer
```

2. Define the traffic limit.

```
[edit firewall policer 1Mbps-policer]
user@host# set if-exceeding bandwidth-limit 1m
user@host# set if-exceeding burst-size-limit 63k
```

   Packets in a traffic flow that conforms to this limit are passed with the PLP set to `low`.

3. Define the actions for nonconforming traffic.

```
[edit firewall policer 1Mbps-policer]
user@host# set then discard
```

   Packets in a traffic flow that exceeds this limit are discarded. Other configurable actions for a single-rate two-color policer are to set the forwarding class and to set the PLP level.

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer 1Mbps-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 63k;
    }
```

```
    then discard;
}
```

Configuring a Prefix-Specific Action Based on the Policer

Step-by-Step Procedure

To configure a prefix-specific action that references the policer and specifies a portion of a source address prefix:

1. Enable configuration of a prefix-specific action.

```
[edit]
user@host# edit firewall family inet prefix-action psa-1Mbps-per-source-24-32-256
```

   Prefix-specific counting and policing can be defined for IPv4 traffic only.

2. Reference the policer for which a prefix-specific set is to be created.

```
[edit firewall family inet prefix-action psa-1Mbps-per-source-24-32-256]
user@host# set policer 1Mbps-policer
user@host# set count
```

> **NOTE**: For aggregated Ethernet interfaces, you can configure a prefix-specific action that references a logical interface policer (also called an aggregate policer). You can reference this type of prefix-specific action from an IPv4 standard firewall filter and then apply the filter at the aggregate level of the interface.

3. Specify the prefix range on which IPv4 addresses are to be indexed to the counter and policer set.

```
[edit firewall family inet prefix-action psa-1Mbps-per-source-24-32-256]
user@host# set source-prefix-length 32
user@host# set subnet-prefix-length 24
```

Results

Confirm the configuration of the prefix-specific action by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer 1Mbps-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 63k;
    }
    then discard;
}
family inet {
    prefix-action psa-1Mbps-per-source-24-32-256 {
        policer 1Mbps-policer;
        subnet-prefix-length 24;
        source-prefix-length 32;
    }
}
```

Configuring an IPv4 Filter That References the Prefix-Specific Action

Step-by-Step Procedure

To configure an IPv4 standard firewall filter that references the prefix-specific action:

1. Enable configuration of the IPv4 standard firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter limit-source-one-24
   ```

   Prefix-specific counting and policing can be defined for IPv4 traffic only.

2. Configure the filter term to match on the packet source address or destination address.

   ```
   [edit firewall family inet filter limit-source-one-24]
   user@host# set term one from source-address 10.10.10.0/24
   ```

3. Configure the filter term to reference the prefix-specific action.

```
[edit firewall family inet filter limit-source-one-24]
user@host# set term one then prefix-action psa-1Mbps-per-source-24-32-256
```

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.

Results

Confirm the configuration of the prefix-specific action by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer 1Mbps-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 63k;
    }
    then discard;
}
family inet {
    prefix-action psa-1Mbps-per-source-24-32-256 {
        policer 1Mbps-policer;
        subnet-prefix-length 24;
        source-prefix-length 32;
    }
    filter limit-source-one-24 {
        term one {
            from {
                source-address {
                    10.10.10.0/24;
                }
            }
            then prefix-action psa-1Mbps-per-source-24-32-256;
        }
    }
}
```

Applying the Firewall Filter to IPv4 Input Traffic at a Logical Interface

Step-by-Step Procedure

To apply the firewall filter to IPv4 input traffic at a logical interface:

1. Enable configuration of IPv4 on the logical interface.

```
[edit]
user@host# edit interfaces so-0/0/2 unit 0 family inet
```

2. Configure an IP address.

```
[edit interfaces so-0/0/2 unit 0 family inet]
user@host# set address 10.39.1.1/16
```

3. Apply the IPv4 standard stateless firewall filter.

```
[edit interfaces so-0/0/2 unit 0 family inet]
user@host# set filter input limit-source-one-24
```

Results

Confirm the configuration of the prefix-specific action by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-0/0/2 {
    unit 0 {
        family inet {
            filter {
                input limit-source-one-24;
            }
            address 10.39.1.1/16;
        }
```

```
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to an Interface | **2096**
- Displaying Prefix-Specific Actions Statistics for the Firewall Filter | **2097**

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to an Interface

Purpose

Verify that the firewall filter `limit-source-one-24` is applied to the IPv4 input traffic at logical interface `so-0/0/2.0`.

Action

Use the `show interfaces statistics` operational mode command for logical interface `so-0/0/2.0`, and include the `detail` option. In the command output section for **Protocol inet**, the **Input Filters** field displays **limit-source-one-24**, indicating that the filter is applied to IPv4 traffic in the input direction:

```
user@host> show interfaces statistics so-0/0/2.0 detail
  Logical interface so-0/0/2.0 (Index 79) (SNMP ifIndex 510) (Generation 149)
    Flags: Hardware-Down Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
    Protocol inet, MTU: 4470, Generation: 173, Route table: 0
      Flags: Sendbcast-pkt-to-re, Protocol-Down
      Input Filters: limit-source-one-24
      Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
        Destination: 10.39/16, Local: 10.39.1.1, Broadcast: 10.39.255.255, Generation: 163
```

Displaying Prefix-Specific Actions Statistics for the Firewall Filter

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show firewall prefix-action-stats filter` *`filter-name`* `prefix-action` *`name`* operational mode command to display statistics about a prefix-specific action configured on a firewall filter.

As an option, you can use the `from` *`set-index`* `to` *`set-index`* command option to specify the starting and ending counter or policer to be displayed. A policer set is indexed from 0 through 65535.

The command output displays the specified filter name followed by a listing of the number of bytes and packets processed by each policer in the policer set.

For a term-specific policer, each policer in the set is identified as follows:

```
prefix-specific-action-name-term-name-set-index
```

For a filter-specific policer, each policer is identified in the command output as follows:

```
prefix-specific-action-name-set-index
```

Because the example prefix-specific action `psa-1Mbps-per-source-24-32-256` is referenced by only one term of the example filter `limit-source-one-24`, the example policer `1Mbps-policer` is configured as term-specific. In the `show firewall prefix-action-stats` command output, the policer statistics are displayed as `psa-1Mbps-per-source-24-32-256-one-0`, `psa-1Mbps-per-source-24-32-256-one-1`, and so on through `psa-1Mbps-per-source-24-32-256-one-255`.

```
user@host> show firewall prefix-action-stats filter limit-source-one-24 prefix-action psa-1Mbps-per-source-24-32-256 from 0 to 9
Filter: limit-source-one-24
Counters:
Name                                                Bytes   Packets
psa-1Mbps-per-source-24-32-256-one-0                    0         0
psa-1Mbps-per-source-24-32-256-one-1                    0         0
psa-1Mbps-per-source-24-32-256-one-2                    0         0
psa-1Mbps-per-source-24-32-256-one-3                    0         0
psa-1Mbps-per-source-24-32-256-one-4                    0         0
```

```
psa-1Mbps-per-source-24-32-256-one-5                0          0
psa-1Mbps-per-source-24-32-256-one-6                0          0
psa-1Mbps-per-source-24-32-256-one-7                0          0
psa-1Mbps-per-source-24-32-256-one-8                0          0
psa-1Mbps-per-source-24-32-256-one-9                0          0
```

SEE ALSO

- Two-Color Policer Configuration Overview | **2034**
- Prefix-Specific Counting and Policing Overview
- Filter-Specific Counter and Policer Set Overview
- Prefix-Specific Counting and Policing Configuration Scenarios

Prefix-Specific Counting and Policing Configuration Scenarios

IN THIS SECTION

- Prefix Length of the Action and Prefix Length of Addresses in Filtered Packets | **2098**
- Scenario 1: Firewall Filter Term Matches on Multiple Addresses | **2100**
- Scenario 2: Subnet Prefix Is Longer Than the Prefix in the Filter Match Condition | **2102**
- Scenario 3: SubnetThe 128th counter and policer Prefix Is Shorter Than the Prefix in the Firewall Filter Match Condition | **2104**

Prefix Length of the Action and Prefix Length of Addresses in Filtered Packets

Table 117 on page 2098 describes the relationship between the prefix length specified in the prefix-specific action and the prefix length of the addresses matched by the firewall filter term that references the prefix-specific action.

Table 117: Summary of Prefix-Specific Action Scenarios

Counter and Policer Set	Packet-Filtering Criteria	Indexing of Instances
Prefix-specific action scenario: Example: Configuring Prefix-Specific Counting and Policing		

Table 117: Summary of Prefix-Specific Action Scenarios *(Continued)*

Counter and Policer Set	Packet-Filtering Criteria	Indexing of Instances	
source-prefix-length = 32 *subnet-prefix-length* = 24 Set size: 2^8 = 256 Instance numbers: 0 - 255	*source-address* = 10.10.10.0/24	Instance 0	10.10.10.0
		Instance 1:	10.10.10.1
		. . .	. . .
		Instance 255:	10.10.10.255
Prefix-specific action scenario: "Scenario 1: Firewall Filter Term Matches on Multiple Addresses" on page 2100			
source-prefix-length = 32 *subnet-prefix-length* = 24 Set size: 2^8 = 256 Instance numbers: 0 - 255	*source-address* = 10.10.10.0/24 *source-address* = 10.11.0.0/16	Instance 0	10.10.10.0, 10.11.*x*.0
		Instance 1:	10.10.10.1, 10.11.*x*.1
		. . .	. . .
		Instance 255:	10.10.10.255, 10.11.*x*.255
	For addresses in the /16 subnet, *x* ranges from 0 through 255.		
Prefix-specific action scenario: "Scenario 2: Subnet Prefix Is Longer Than the Prefix in the Filter Match Condition" on page 2102			
source-prefix-length = 32 *subnet-prefix-length* = 25 Set size: 2^7 = 128 Instance numbers: 0 - 127	*source-address* = 10.10.10.0/24	Instance 0	10.10.10.0, 10.10.10.128
		Instance 1:	10.10.10.1, 10.10.10.120

Table 117: Summary of Prefix-Specific Action Scenarios *(Continued)*

Counter and Policer Set	Packet-Filtering Criteria	Indexing of Instances	
		. . .	. . .
		Instance 127:	10.10.10.255, 10.10.10.127
Prefix-specific action scenario: "Scenario 3: SubnetThe 128th counter and policer Prefix Is Shorter Than the Prefix in the Firewall Filter Match Condition" on page 2104			
source-prefix-length = 32 *subnet-prefix-length* = 24 Set size: 2^8 = 256 Instance numbers: 0 - 255	*source-address* = 10.10.10.0/25 **NOTE**: Only packets with source addresses ranging from `10.10.10.0` through `10.10.10.127` are passed to the prefix-specific action.	Instance 0	10.10.10.0
		Instance 1:	10.10.10.1
		. . .	. . .
		Instance 127:	10.10.10.127
		Instances 128 – 255: unused	

Scenario 1: Firewall Filter Term Matches on Multiple Addresses

The complete example, Example: Configuring Prefix-Specific Counting and Policing, shows the simplest case of prefix-specific actions, in which a single-term *firewall filter* matches on one address with a prefix length that is the same as the subnet prefix length specified in the prefix-specific action. Unlike the example, this scenario describes a configuration in which a single-term firewall filter matches on two IPv4 source addresses. In addition, the additional condition matches on a source address with a prefix length that is different from the subnet prefix length defined in the prefix-specific action. In this case, the additional condition matches on the `/16` subnet of the source address `10.11.0.0`.

NOTE: Unlike packets that match the source address `10.10.10.0/24`, packets that match the source address `10.11.0.0/16` are in a many-to-one correspondence with the instances in the counter and policer set.

The filter-matched packets that are passed to the prefix-specific action index into the counter and policer set in such a way that the counting and policing instances are shared by packets that contain source addresses across the 10.10.10.0/24 and 10.11.0.0/16 subnets as follows:

- The first counter and policer in the set are indexed by packets with source addresses 10.10.10.0 and 10.11.*x*.0, where *x* ranges from 0 through 255.
- The second counter and policer in the set are indexed by packets with source addresses 10.10.10.1 and 10.11.*x*.1, where *x* ranges from 0 through 255.
- The 256th (last) counter and policer in the set are indexed by packets with source addresses 10.10.10.255 and 10.11.*x*.255, where *x* ranges from 0 through 255.

The following configuration shows the statements for configuring the single-rate two-color policer, the prefix-specific action that references the policer, and the IPv4 standard stateless firewall filter that references the prefix-specific action:

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
        then discard;
    }
    family inet {
        prefix-action psa-1Mbps-per-source-24-32-256 {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
        filter limit-source-two-24-16 {
            term one {
                from {
                    source-address {
                        10.10.10.0/24;
                        10.11.0.0/16;
                    }
                }
                then prefix-action psa-1Mbps-per-source-24-32-256;
            }
        }
```

```
    }
}
interfaces {
    so-0/0/2 {
        unit 0 {
            family inet {
                filter {
                    input limit-source-two-24-16;
                }
                address 10.39.1.1/16;
            }
        }
    }
}
```

Scenario 2: Subnet Prefix Is Longer Than the Prefix in the Filter Match Condition

The complete example, Example: Configuring Prefix-Specific Counting and Policing, shows the simplest case of prefix-specific actions, in which the single-term firewall filter matches on one address with a prefix length that is the same as the subnet prefix length specified in the prefix-specific action. Unlike the example, this scenario describes a configuration in which the prefix-specific action defines a subnet prefix length that is longer than the prefix of the source address matched by the firewall filter. In this case, the prefix-specific action defines a subnet-prefix value of 25, while the firewall filter matches on a source address in the /24 subnet.

NOTE: The firewall filter passes the prefix-specific action packets with source addresses that range from 10.10.10.0 through 10.10.10.255, while the prefix-specific action specifies a set of only 128 counters and policers, numbered from 0 through 127.

The filter-matched packets that are passed to the prefix-specific action index into the counter and policer set in such a way that the counting and policing instances are shared by packets that contain either of two source addresses within the 10.10.10.0/24 subnet:

- The first counter and policer in the set are indexed by packets with source addresses 10.10.10.0 and 10.10.10.128.
- The second counter and policer in the set are indexed by packets with source addresses 10.10.10.1 and 10.10.10.129.
- The 128th (last) counter and policer in the set are indexed by packets with source addresses 10.10.10.127 and 10.10.10.255.

The following configuration shows the statements for configuring the single-rate two-color policer, the prefix-specific action that references the policer, and the IPv4 standard stateless firewall filter that references the prefix-specific action:

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
        then discard;
    }
    family inet {
        prefix-action psa-1Mbps-per-source-25-32-128 {
            policer 1Mbps-policer;
            subnet-prefix-length 25;
            source-prefix-length 32;
        }
        filter limit-source-one-24 {
            term one {
                from {
                    source-address {
                        10.10.10.0/24;
                    }
                }
                then prefix-action psa-1Mbps-per-source-25-32-128;
            }
        }
    }
}
interfaces {
    so-0/0/2 {
        unit 0 {
            family inet {
                filter {
                    input limit-source-one-24;
                }
                address 10.39.1.1/16;
            }
        }
```

```
    }
}
```

Scenario 3: SubnetThe 128th counter and policer Prefix Is Shorter Than the Prefix in the Firewall Filter Match Condition

The complete example, Example: Configuring Prefix-Specific Counting and Policing, shows the simplest case of prefix-specific actions, in which the single-term firewall filter matches on one address with a prefix length that is the same as the subnet prefix length specified in the prefix-specific action. Unlike the example, this scenario describes a configuration in which the prefix-specific action defines a subnet prefix length that is shorter than the prefix of the source address matched by the firewall filter. In this case, the filter term matches on the `/25` subnet of the source address `10.10.10.0`.

NOTE: The firewall filter passes the prefix-specific action only packets with source addresses that range from `10.10.10.0` through `10.10.10.127`, while the prefix-specific action specifies a set of 256 counters and policers, numbered from 0 through 255.

The matched packets that are passed to the prefix-specific action index into the lower half of the counter and policer set only:

- The first counter and policer in the set are indexed by packets with source address `10.10.10.0`.
- The second counter and policer in the set are indexed by packets with source address `10.10.10.1` and `10.10.10.129`.
- The 128th counter and policer in the set are indexed by packets with source address `10.10.10.127`.
- The upper half of the set (instances numbered from 128 through 255) are not indexed by packets passed to the prefix-specific action from this particular firewall filter.

The following configuration shows the statements for configuring the single-rate two-color policer, the prefix-specific action that references the policer, and the IPv4 standard stateless firewall filter that references the prefix-specific action:

```
[edit]
firewall {
    policer 1Mbps-policer {
        if-exceeding {
            bandwidth-limit 1m;
            burst-size-limit 63k;
        }
        then discard;
```

```
    }
    family inet {
        prefix-action psa-1Mbps-per-source-24-32-256 {
            policer 1Mbps-policer;
            subnet-prefix-length 24;
            source-prefix-length 32;
        }
        filter limit-source-one-25 {
            term one {
                from {
                    source-address {
                        10.10.10.0/25;
                    }
                }
                then prefix-action psa-1Mbps-per-source-24-32-256;
            }
        }
    }
}
interfaces {
    so-0/0/2 {
        unit 0 {
            family inet {
                filter {
                    input limit-source-one-25;
                }
                address 10.39.1.1/16;
            }
        }
    }
}
```

SEE ALSO

- Two-Color Policer Configuration Overview | **2034**
- Prefix-Specific Counting and Policing Overview
- Filter-Specific Counter and Policer Set Overview
- Prefix-Specific Counting and Policing Configuration Scenarios

RELATED DOCUMENTATION

Two-Color Policer Configuration Overview | 2034

Guidelines for Applying Traffic Policers | 1934

Policer Overhead to Account for Rate Shaping in the Traffic Manager

IN THIS SECTION

- Policer Overhead to Account for Rate Shaping Overview | 2106
- Example: Configuring Policer Overhead to Account for Rate Shaping | 2107

Policer Overhead to Account for Rate Shaping Overview

If you configure ingress or egress traffic-shaping overhead values for an interface, the traffic manager cannot apply these values to any rate-limiting also applied to the interface. To enable the router to account for the additional Ethernet frame length when policing actions are being determined, you must configure the ingress or egress overhead values for policers separately.

NOTE: When a policer overhead value is changed, the PIC or DPC goes offline and then comes back online.

For Gigabit Ethernet Intelligent Queuing 2 (IQ2) and Enhanced IQ2 (IQ2E) PICs or interfaces on Dense Port Concentrators (DPCs) in MX Series routers, you can control the rate of traffic that passes through all interfaces on the PIC or DPC by configuring a *policer overhead*. You can configure a policer ingress overhead and a policer egress overhead, each with values from 0 through 255 bytes. The policer overhead values are added to the length of the final Ethernet frame when determining ingress and egress policer actions.

SEE ALSO

egress-policer-overhead

ingress-policer-overhead

Example: Configuring Policer Overhead to Account for Rate Shaping

IN THIS SECTION

- Requirements | 2107
- Overview | 2107
- Configuration | 2108
- Verification | 2116

This example shows how to configure overhead values for policers when rate-shaping overhead is configured.

Requirements

Before you begin, make sure that interface for which you are applying ingress or egress policer overhead is hosted on one of the following:

- Gigabit Ethernet IQ2 PIC
- IQ2E PIC
- DPCs in MX Series routers

Overview

IN THIS SECTION

- Topology | 2108

This example shows how to configure policer overhead values for all physical interfaces on a supported PIC or MPC so that the rate shaping value configured on a logical interface is accounted for in any policing on that logical interface.

Topology

The router hosts a Gigabit Ethernet IQ2 PIC, installed in PIC location 3 of the Flexible PIC Concentrator (FPC) in slot number 1. The physical interface on port 1 on that PIC is configured to receive traffic on logical interface 0 and send it back out on logical interface 1. Class-of-service scheduling includes 100 Mbps of traffic rate-shaping overhead for the output traffic. A policer egress overhead of 100 bytes is configured on the entire PIC so that, for any policers applied to the output traffic, 100 bytes are added to the final Ethernet frame length when determining ingress and egress policer actions.

NOTE: Traffic rate-shaping and corresponding policer overhead are configured separately:

- You configure rate shaping at the `[edit class-of-service interfaces interface-name unit unit-number]` hierarchy level.
- You configure policer overhead at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level.

When a policer overhead value is changed, the PIC or DPC goes offline and then comes back online.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2109**
- Configuring the Logical Interfaces | **2109**
- Configuring Traffic Rate-Shaping on the Logical Interface That Carries Output Traffic | **2111**
- Configuring Policer Overhead on the PIC or DPC That Hosts the Rate-Shaped Logical Interface | **2113**
- Applying a Policer to the Logical Interface That Carries Input Traffic | **2114**

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

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces ge-1/3/1 per-unit-scheduler
set interfaces ge-1/3/1 vlan-tagging
set interfaces ge-1/3/1 unit 0 vlan-id 100
set interfaces ge-1/3/1 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/3/1 unit 1 vlan-id 101
set interfaces ge-1/3/1 unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac
00:00:11:22:33:44
set class-of-service schedulers be transmit-rate percent 5
set class-of-service schedulers ef transmit-rate percent 30
set class-of-service schedulers af transmit-rate percent 30
set class-of-service schedulers nc transmit-rate percent 35
set class-of-service scheduler-maps my-map forwarding-class best-effort scheduler be
set class-of-service scheduler-maps my-map forwarding-class expedited-forwarding scheduler ef
set class-of-service scheduler-maps my-map forwarding-class network-control scheduler nc
set class-of-service scheduler-maps my-map forwarding-class assured-forwarding scheduler af
set class-of-service interfaces ge-1/3/1 unit 1 scheduler-map my-map
set class-of-service interfaces ge-1/3/1 unit 1 shaping-rate 100m
set firewall policer 500Kbps logical-interface-policer
set firewall policer 500Kbps if-exceeding bandwidth-limit 500k
set firewall policer 500Kbps if-exceeding burst-size-limit 625k
set firewall policer 500Kbps then discard
set chassis fpc 1 pic 3 ingress-policer-overhead 100
set chassis fpc 1 pic 3 egress-policer-overhead 100
set interfaces ge-1/3/1 unit 0 family inet policer input 500Kbps
```

Configuring the Logical Interfaces

Step-by-Step Procedure

To configure the logical interfaces:

1. Enable configuration of the interface

```
[edit]
user@host# edit interfaces ge-1/3/1
```

2. Enable multiple queues for each logical interface (so that you can associate an output scheduler with each logical interface).

```
[edit interfaces ge-1/3/1]
user@host# set per-unit scheduler
user@host# set vlan-tagging
```

> **NOTE**: For Gigabit Ethernet IQ2 PICs only, use the `shared-scheduler` statement to enable shared schedulers and shapers on a physical interface.

3. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 0 vlan-id 100
user@host# set unit 0 family inet address 10.10.10.1/30
```

4. Configure logical interface `ge-1/3/1.1`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 1 vlan-id 101
user@host# set unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac 00:00:11:22:33:44
```

Results

Confirm the configuration of the interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    per-unit-scheduler;
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.10.10.1/30;
```

```
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

Configuring Traffic Rate-Shaping on the Logical Interface That Carries Output Traffic

Step-by-Step Procedure

To configure traffic rate-shaping on the logical interface that carries output traffic:

1. Enable configuration of class-of-service features.

```
[edit]
user@host# edit class-of-service
```

2. Configure packet scheduling on logical interface `ge-1/3/1.0`.

   - Configure schedulers that specify the percentage of transmission capacity.

```
[edit class-of-service]
user@host# edit schedulers

[edit class-of-service schedulers]
user@host# set be transmit-rate percent 5
user@host# set ef transmit-rate percent 30
user@host# set af transmit-rate percent 30
user@host# set nc transmit-rate percent 35
```

   A percentage of zero drops all packets in the queue. When the `rate-limit` option is specified, the transmission rate is limited to the rate-controlled amount. In contrast with the `exact` option, a scheduler with the `rate-limit` option shares unused bandwidth above the rate-controlled amount.

- Configure a scheduler map to associate each scheduler with a forwarding class.

```
[edit class-of-service]
user@host# edit scheduler-maps my-map

[edit class-of-service scheduler-maps my-map]
user@host# set forwarding-class best-effort scheduler be
user@host# set forwarding-class expedited-forwarding scheduler ef
user@host# set forwarding-class network-control scheduler nc
user@host# set forwarding-class assured-forwarding scheduler af
```

- Associate the scheduler map with logical interface `ge-1/3/1.0`.

```
[edit class-of-service]
user@host# edit interfaces ge-1/3/1 unit 1

[edit class-of-service interfaces ge-1/3/1 unit 1]
user@host# set scheduler-map my-map
```

3. Configure 100 Mbps of traffic rate-shaping overhead on logical interface `ge-1/3/1.1`.

```
[edit class-of-service interfaces ge-1/3/1 unit 1]
user@host# set shaping-rate 100
```

Alternatively, you can configure a shaping rate for a logical interface and oversubscribe the physical interface by including the `shaping-rate` statement at the `[edit class-of-service traffic-control-profiles]` hierarchy level. With this configuration approach, you can independently control the delay-buffer rate.

Results

Confirm the configuration of the class-of-service features (including the 100 Mbp of shaping of the egress traffic) by entering the `show class-of-service` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show class-of-service
interfaces {
```

```
    ge-1/3/1 {
        unit 1 {
            scheduler-map my-map;
            shaping-rate 100m;
        }
    }
}
scheduler-maps {
    my-map {
        forwarding-class best-effort scheduler be;
        forwarding-class expedited-forwarding scheduler ef;
        forwarding-class network-control scheduler nc;
        forwarding-class assured-forwarding scheduler af;
    }
}
schedulers {
    be {
        transmit-rate percent 5;
    }
    ef {
        transmit-rate percent 30;
    }
    af {
        transmit-rate percent 30;
    }
    nc {
        transmit-rate percent 35;
    }
}
```

Configuring Policer Overhead on the PIC or DPC That Hosts the Rate-Shaped Logical Interface

Step-by-Step Procedure

To configure policer overhead on the PIC or MPC that hosts the rate-shaped logical interface:

1. Enable configuration of the supported PIC or MPC.

```
[edit]
user@host# set chassis fpc 1 pic 3
```

2. Configure 100 bytes of policer overhead on the supported PIC or MPC.

```
[edit chassis fpc 1 pic 3]
user@host# set ingress-policer-overhead 100
user@host# set egress-policer-overhead 100
```

> **NOTE**: These values are added to the length of the final Ethernet frame when determining ingress and egress policer actions for all physical interfaces on the PIC or MPC.

You can specify policer overhead with values from 0 through 255 bytes.

Results

Confirm the configuration of the policer overhead on the physical interface to account for rate-shaping by entering the `show chassis` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show chassis
chassis {
    fpc 1 {
        pic 3 {
            egress-policer-overhead 100;
            ingress-policer-overhead 100;
        }
    }
}
```

Applying a Policer to the Logical Interface That Carries Input Traffic

Step-by-Step Procedure

To apply a policer to the logical interface that carries input traffic:

1. Configure the logical interface (aggregate) policer.

```
[edit]
user@host# edit firewall policer 500Kbps
```

```
[edit firewall policer 500Kbps]
user@host# set logical-interface-policer
user@host# set if-exceeding bandwidth-limit 500k
user@host# set if-exceeding burst-size-limit 625k
user@host# set then discard
```

2. Apply the policer to Layer 3 input on the IPv4 logical interface.

```
[edit]
user@host# set interfaces ge-1/3/1 unit 0 family inet policer input 500Kbps
```

NOTE: The 100 Mbps policer overhead is added to the length of the final Ethernet frame when determining ingress and egress policer actions,

Results

Confirm the configuration of the policer with rate-shaping overhead by entering the `show firewall` and `show interfaces` configuration mode commands. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer 500Kbps {
    logical-interface-policer;
    if-exceeding {
        bandwidth-limit 500k;
        burst-size-limit 625k;
    }
    then discard;
}

[edit]
user@host# show interfaces
ge-1/3/1 {
    per-unit-scheduler;
    vlan-tagging;
    unit 0 {
```

```
        vlan-id 100;
        layer2-policer {
            input-policer 500Kbps;
        }
        family inet {
            address 10.10.10.1/30;
        }
    }
    unit 0 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | **2116**
- Displaying Statistics for the Policer | **2117**

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interface `ge-1/3/1.0`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface, and the **Protocol inet** section contains a **Policer** field that would list the policer `500Kbps` as an input or output policer as follows:

- **Input: 500Kbps-ge-1/3/1.0-log_int-i**
- **Output: 500Kbps-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to Input traffic only.

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction. For the policer `500Kbps`, the input and output policer names are displayed as follows:

- **500Kbps-ge-1/3/1.0-log_int-i**
- **500Kbps-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to input traffic only.

SEE ALSO

> *egress-policer-overhead*
>
> *ingress-policer-overhead*

RELATED DOCUMENTATION

Two-Color Policer Configuration Overview | 2034

Guidelines for Applying Traffic Policers | 1934

Configuring a Policer Overhead | 2001

CLI Explorer

Three-Color Policer Configuration Overview

Table 118 on page 2118 describes the hierarchy levels at which you can configure and apply single-rate tricolor-marking (single-rate TCM) policers and two-rate tricolor-marking (two-rate TCM) policers to Layer 3 traffic. For information about applying three-color policers to Layer 2 traffic, see Three-Color Policing at Layer 2 Overview.

Table 118: Three-Color Policer Configuration and Application Overview

Policer Configuration	Layer 3 Application	Key Points
Single-Rate Three-Color Policer	Defines traffic rate limiting that you can apply to Layer 3 protocol-specific traffic at a logical interface. Can be applied as a firewall filter policer only.	Provides moderate allowances for short periods of traffic that exceed the committed burst size.

Table 118: Three-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Basic single-rate TCM policer configuration:
 	
`[edit firewall]`
`three-color-policer` *`policer-name`* `{`
`  single-rate {`
`  (color-aware | color-blind);`
`  committed-information-rate` *`bps`*`;`
`  committed-burst-size` *`bytes`*`;`
`  excess-burst-size` *`bytes`*`;`
`  }`
`  action {`
`  loss-priority high then discard;`
`  }`
`}`
	Reference the policer from a firewall filter, and apply the filter to a protocol family on a logical interface:
	
	`[edit firewall]`
	`family` *`family-name`* `{`
	`  filter` *`filter-name`* `{`
	`  term` *`term-name`* `{`
	`  from {`
	`  ... match-conditions ...`
	`  }`
	`  then {`
	`  three-color-policer {`
	`  single-rate` *`policer-name`*`;`
	`  }`
	`  }`
	`  }`
	`  }`
	`}`
	
	Apply the filter to a logical interface at the protocol family level:
	
	`[edit interfaces]`
	`interface-name` `{`
	`  unit` *`unit-number`* `{`
	`  family` *`family-name`* `{`
	`  filter {`
	`  input` *`filter-name`*`;`
	`  output` *`filter-name`*`;`
	`  }`
	`  }`
	`  }`
	`}`
		Policer configuration:
		
		• Include the `single-rate (color-aware | color-blind)` statement.
		
		Firewall filter configuration:
		
		• Include the `three-color-policer single-rate` *`policer-name`* action.
		
		Applying the firewall filter to the logical interface:
		
		• Include the `filter (input | output)` *`filter-name`* statement.

Single-Rate Three-Color Physical Interface Policer

Defines traffic rate limiting that applies to all logical interfaces and protocol families configured on a physical interface, even if the interfaces belong to different routing instances. Can be applied as a firewall filter policer only.

Table 118: Three-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Physical interface single-rate TCM policer:
 	
`[edit firewall]`
`three-color-policer` *`policer-name`* `{`
`  physical-interface-policer;`
`  single-rate {`
`  (color-aware | color-blind);`
`  committed-information-rate` *`bps`*`;`
`  committed-burst-size` *`bytes`*`;`
`  excess-burst-size` *`bytes`*`;`
`  }`
`  action {`
`  loss-priority high then discard;`
`  }`
`}`
	Reference the policer from a physical interface filter only, and apply the filter to a protocol family on a logical interface:
	
	`[edit firewall]`
	`family` *`family-name`* `{`
	`  filter` *`filter-name`* `{`
	`  physical-interface-filter`
	`  term` *`term-name`* `{`
	`  from {`
	`  ... match-conditions ...`
	`  }`
	`  then {`
	`  three-color-policer {`
	`  single-rate` *`policer-name`*`;`
	`  }`
	`  }`
	`  }`
	`  }`
	`}`
	
	`[edit interfaces]`
	`interface-name` `{`
	`  unit` *`number`* `{`
	`  family` *`family-name`* `{`
	`  filter {`
	`  input` *`filter-name`*`;`
	`  output` *`filter-name`*`;`
	`  }`
	`  }`
	`  }`
	`}`
		Policer configuration:
		
		• Include the `physical-interface-policer` statement.
		
		Firewall filter configuration:
		
		• Include the `physical-interface-filter` statement.
		
		Application:
		
		• Include the `filter (input | output)` *`filter-name`* statement.
		
		Verification
		
		• To verify, use the `show firewall filter` *`filter-name`* operational mode command.

Basic Two-Rate Three-Color Policer

Defines traffic rate limiting that you can apply to Layer 3 protocol-specific traffic at a logical interface. Can be applied as a firewall filter policer only.

Provides moderate allowances for sustained periods of traffic that exceed the committed bandwidth limit or burst size.

Table 118: Three-Color Policer Configuration and Application Overview *(Continued)*

Policer Configuration	Layer 3 Application	Key Points

Basic two-rate TCM policer configuration:
 	
`[edit firewall]`
`three-color-policer` *`policer-name`* `{`
`  two-rate {`
`  (color-aware | color-blind);`
`  committed-information-rate` *`bps`*`;`
`  committed-burst-size` *`bytes`*`;`
`  peak-information-rate` *`bps`*`;`
`  peak-burst-size` *`bytes`*`;`
`  }`
`  action {`
`  loss-priority high then discard;`
`  }`
`}`
	Reference the policer from a firewall filter, and apply the filter to a protocol family on a logical interface:
	
	`[edit firewall]`
	`family` *`family-name`* `{`
	`  filter` *`filter-name`* `{`
	`  term` *`term-name`* `{`
	`  from {`
	`  ...` *`match-conditions`* `...`
	`  }`
	`  then {`
	`  three-color-policer {`
	`  two-rate` *`policer-name`*`;`
	`  }`
	`  }`
	`  }`
	`  }`
	`}`
	
	`[edit interfaces]`
	`interface-name` `{`
	`  unit` *`unit-number`* `{`
	`  family` *`family-name`* `{`
	`  filter {`
	`  input` *`filter-name`*`;`
	`  output` *`filter-name`*`;`
	`  }`
	`  }`
	`  }`
	`}`
		Policer configuration:
		
		• Include the `two-rate (color-aware | color-blind)` statement.
		
		Firewall filter configuration:
		
		• Include the `three-color-policer two-rate` *`policer-name`* action.
		
		Applying the firewall filter to the logical interface:
		
		• Include the `filter (input | output)` *`filter-name`* statement.

RELATED DOCUMENTATION

Three-Color Policer Configuration Guidelines | 2134

Basic Single-Rate Three-Color Policers | 2138

Basic Two-Rate Three-Color Policers | 2147

Two-Color and Three-Color Logical Interface Policers | 2166

Two-Color and Three-Color Physical Interface Policers | 2185

Applying Policers

IN THIS SECTION

- Overview of Applying Policers | 2122
- Applying Aggregate Policers | 2123
- Applying Hierarchical Policers on Enhanced Intelligent Queuing PICs | 2126
- Configuring Hierarchical Policers | 2129
- Configuring a Single-Rate Two-Color Policer | 2130
- Configuring a Single-Rate Color-Blind Policer | 2131
- Configuring a Two-Rate Tricolor Marker Policer | 2132

Overview of Applying Policers

Policers allow you to perform simple traffic policing on specific interfaces or Layer 2 virtual private networks (VPNs) without configuring a firewall filter. To apply policers, include the `policer` statement:

```
policer {
    arp policer-template-name;
    input policer-template-name;
    output policer-template-name;
}
```

You can include these statements at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number family family]`
- `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number family family]`

In the `family` statement, the protocol family can be `ccc`, `inet`, `inet6`, `mpls`, `tcc`, or `vpls`.

In the `arp` statement, list the name of one policer template to be evaluated when Address Resolution Protocol (ARP) packets are received on the interface. By default, an ARP policer is installed that is shared

among all the Ethernet interfaces on which you have configured the `family inet` statement. If you want more stringent or lenient policing of ARP packets, you can configure an interface-specific policer and apply it to the interface. You configure an ARP policer just as you would configure any other policer, at the `[edit firewall policer]` hierarchy level. If you apply this policer to an interface, the default ARP packet policer is overridden. If you delete this policer, the default policer takes effect again.

- You can configure a different policer on each protocol family on an interface, with one input policer and one output policer for each family. When you apply policers, you can configure the family `ccc`, `inet`, `inet6`, `mpls`, `tcc`, or `vpls` only, and one ARP policer for the family `inet` protocol only. Each time a policer is referenced, a separate copy of the policer is installed on the packet forwarding components for that interface.
- If you apply both policers and firewall filters to an interface, input policers are evaluated before input firewall filters, and output policers are evaluated after output firewall filters. In the `input` statement, list the name of one policer template to be evaluated when packets are received on the interface. In the `output` statement, list the name of one policer template to be evaluated when packets are transmitted on the interface.
- For subscribers terminating on MX Series routers over an Aggregated Ethernet (AE) interface that spans multiple FPCs, it is possible for an overall subscriber rate to exceed the configured rate because the limit configured in the policer is applied separately to each interface in the AE bundle. Thus, for example, if you intend to have a policer on a three-member AE interface enforce a `bandwidth-limit` of *600m*, you would need to configure the `bandwidth-limit` in the policer for *200m* to account for the three interfaces in the AE (that is, 200 Mbps per interface, for a combined total of 600Mbps).
- If you apply the policer to the interface `lo0`, it is applied to packets received or transmitted by the Routing Engine.
- On T Series, M120, and M320 platforms, if the interfaces are on the same FPC, the filters or policers do not act on the sum of traffic entering and exiting the interfaces.

Applying Aggregate Policers

IN THIS SECTION

- Applying Aggregate Policers | **2123**

Applying Aggregate Policers

By default, if you apply a policer to multiple protocol families on the same logical interface, the policer restricts traffic for each protocol family individually. For example, a policer with a 50 Mbps bandwidth

limit applied to both IPv4 and IPv6 traffic would allow the interface to accept 50 Mbps of IPv4 traffic and 50 Mbps of IPv6 traffic. If you apply an aggregate policer, the policer would allow the interface to receive only 50 Mbps of IPv4 and IPv6 traffic combined.

To configure an aggregate policer, include the `logical-interface-policer` statement at the `[edit firewall policer policer-template-name]` hierarchy level:

```
[edit firewall policer policer-template-name]
logical-interface-policer;
```

For the policer to be treated as an aggregate, you must apply it to multiple protocol families on a single logical interface by including the `policer` statement:

```
policer {
    arp policer-template-name;
    input policer-template-name;
    output policer-template-name;
}
```

You can include these statements at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number family family]`
- `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number family family]`

In the `family` statement, the protocol family can be `ccc`, `inet`, `inet6`, `mpls`, `tcc`, or `vpls`.

The protocol families on which you do not apply the policer are not affected by the policer. For example, if you configure a single logical interface to accept MPLS, IPv4, and IPv6 traffic and you apply the logical interface policer `policer1` to only the IPv4 and IPv6 protocol families, MPLS traffic is not subject to the constraints of `policer1`.

If you apply `policer1` to a different logical interface, there are two instances of the policer. This means the Junos OS polices traffic on separate logical interfaces separately, not as an aggregate, even if the same logical-interface policer is applied to multiple logical interfaces on the same physical interface port.

Example: Applying Aggregate Policers

Configure two logical interface policers: `aggregate_police1` and `aggregate_police2`. Apply `aggregate_police1` to IPv4 and IPv6 traffic received on logical interface `fe-0/0/0.0`. Apply `aggregate_police2` to CCC and MPLS traffic received on logical interface fe-0/0/0.0. This configuration causes the software to create only one instance of `aggregate_police1` and one instance of `aggregate_police2`.

Apply aggregate_police1 to IPv4 and IPv6 traffic received on another logical interface fe-0/0/0.1. This configuration causes the software to create a new instance of aggregate_police1, one that applies to unit 0 and another that applies to unit 1.

```
[edit firewall]
policer aggregate_police1 {
    logical-interface-policer;
    if-exceeding {
        bandwidth-limit 100m;
        burst-size-limit 500k;
    }
    then {
        discard;
    }
}
policer aggregate_police2 {
    logical-interface-policer;
    if-exceeding {
        bandwidth-limit 10m;
        burst-size-limit 200k;
    }
    then {
        discard;
    }
}
[edit interfaces fe-0/0/0]
unit 0 {
    family inet {
        policer {
            input aggregate_police1;
        }
    }
    family inet6 {
        policer {
            input aggregate_police1;
        }
    }
    family ccc {
        policer {
            input aggregate_police2;
        }
    }
```

```
    family mpls {
        policer {
            input aggregate_police2;
        }
    }
}
unit 1 {
    family inet {
        policer {
            input aggregate_police1;
        }
    }
    family inet6 {
        policer {
            input aggregate_police1;
        }
    }
}
```

Applying Hierarchical Policers on Enhanced Intelligent Queuing PICs

IN THIS SECTION

- Applying Hierarchical Policers on Enhanced Intelligent Queuing PICs | **2126**

Applying Hierarchical Policers on Enhanced Intelligent Queuing PICs

M40e, M120, and M320 edge routers and T Series core routers with Enhanced Intelligent Queuing (IQE) PICs support hierarchical policers in the ingress direction and allow you to apply a hierarchical policer for the premium and aggregate (premium plus normal) traffic levels to an interface. Hierarchical policers provide cross-functionality between the configured physical interface and the Packet Forwarding Engine.

Before you begin, there are some general restrictions that apply to hierarchical policers:

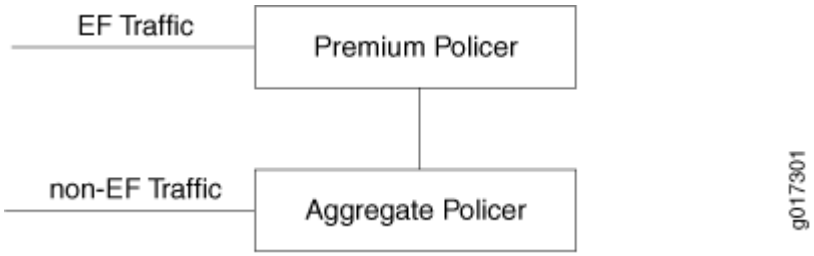
- Only one type of policer can be configured for a logical or physical interface. For example, a hierarchical policer and a regular policer in the same direction for the same logical interface is not allowed.

- The chaining of the policers—that is, applying policers to both a port and the logical interfaces of that port—is not allowed.
- There is a limit of 64 policers per interface in case there is no BA classification, providing a single policer per DLCI.
- Only one kind of policer can be applied on a physical or logical interface.
- The policer should be independent of BA classification. Without BA classification, all traffic on an interface will be treated either as EF or non-EF, based on the configuration. With BA classification, an interface can support up to 64 policers. Again, the interface here may be a physical interface or logical interface (for example, DLCI).
- With BA classification, the miscellaneous traffic (the traffic *not* matching with any of the BA classification DSCP/EXP bits) will be policed as non-EF traffic. No separate policers will be installed for this traffic.

Hierarchical Policer Overview

Hierarchical policing uses two token buckets, one for aggregate (non-EF) traffic and one for premium (EF) traffic. Which traffic is EF and which is non-EF is determined by the class-of-service configuration. Logically, hierarchical policing is achieved by chaining two policers.

Figure 88: Hierarchical Policer



In the example in Figure 88 on page 2127, EF traffic is policed by Premium Policer and non EF traffic is policed by Aggregate Policer. What that means is, for EF traffic the out-of-spec action will be the one that is configured for Premium Policer, but the in-spec EF traffic will still consume the tokens from the Aggregate Policer.

But EF traffic will never be submitted to the out-of-spec action of the Aggregate Policer. Also, if the out-of-spec action of the Premium Policer is not set to Discard, those out-of-spec packets will not consume the tokens from the Aggregate Policer. Aggregate Policer only polices the non-EF traffic. As you can see, the Aggregate Policer token bucket can go negative, if all the tokens are consumed by the non-EF traffic and then you get bursts of EF traffic. But that will be for a very short time, and over a period of time it will average out. For example:

- *Premium Policer:* Bandwidth 2 Mbps, OOS Action: Discard
- *Aggregate Policer:* Bandwidth 10 Mbps, OOS Action: Discard

In the above case, EF traffic is guaranteed 2 Mbps and the non-EF traffic will get from 8 Mbps to 10 Mbps, depending on the input rate of the EF traffic.

Hierarchical Policing Characteristics

Hierarchical token bucket features include:

- Ingress traffic is first classified into EF and non-EF traffic prior to applying a policer:
  - Classification is performed by Q-tree lookup
- Channel number selects a shared token bucket policer:
  - Dual token bucket policer is divided into two single bucket policers:
    - Policer1—EF traffic
    - Policer2—non-EF traffic
- Shared token bucket is used to police the traffic as follows:
  - Policer1 is set to EF rate (for example, 2 Mbps)
  - Policer2 is set to aggregate interface policed rate (for example, 10 Mbps).
  - EF traffic gets applied to Policer1.
    - If traffic is in-spec it is allowed to pass and decrement from both Policer1 and Policer2.
    - If traffic is out-of-spec it can be discarded or marked with a new FC or loss priority. Policer2 will not do anything with out-of-spec EF traffic.
  - Non-EF traffic gets applied only to Policer2.
    - If traffic is in-spec it is allowed to pass through and decremented Policer2.
    - If traffic is out-of-spec it is discarded or marked with a new FC or set with a new drop priority.
- Rate-limit the port speed to a desired rate at Layer 2
- Rate-limit the EF traffic
- Rate-limit the non-EF traffic
- Policing drops counted per color

SEE ALSO

Class of Service User Guide (Routers and EX9200 Switches)

Configuring Hierarchical Policers

To configure a hierarchical policer, apply the `policing-priority` statement to the proper forwarding class and configure a hierarchical policer for the aggregate and premium level. For more information about class of service, see the Junos OS Class of Service User Guide for Routing Devices.

NOTE: Hierarchical policers can only be configured on SONET physical interfaces hosted on an IQE PIC. Only aggregate and premium levels are supported.

CoS Configuration of Forwarding Classes for Hierarchical Policers

```
[edit class-of-service forwarding-classes]
class fc1 queue-num 0 priority high policing-priority premium;
class fc2 queue-num 1 priority low policing-priority normal;
class fc3 queue-num 2 priority low policing-priority normal;
class fc4 queue-num 3 priority low policing-priority normal;
```

For detailed information on class-of-service configuration and statements, see the Junos OS Class of Service User Guide for Routing Devices.

Firewall Configuration for Hierarchical Policers

```
[edit firewall hierarchical-policer foo]
aggregate {
    if-exceeding {
        bandwidth-limit 70m;
        burst-size-limit 1500;
    }
    then {
    discard;
}
premium {
    if-exceeding {
        bandwidth-limit 50m;
        burst-size-limit 1500;
    }
    then {
```

```
        discard;
    }
}
```

You can apply the hierarchical policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-hierarchical-policer foo;
```

You also have the option to apply the policer at the physical port level as follows:

```
[edit interfaces so-0/1/0 layer2-policer]
input-hierarchical-policer foo;
```

Configuring a Single-Rate Two-Color Policer

You can configure a single-rate two-color policer as follows:

```
[edit firewall policer foo]
    if-exceeding {
        bandwidth-limit 50m;
        burst-size-limit 1500;
    }
then {
    discard;
}
```

You can apply the policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-policer foo;
```

You also have the option to apply the policer at the physical port level as follows:

```
[edit interfaces so-0/1/0 layer2-policer]
input-policer foo;
```

Configuring a Single-Rate Color-Blind Policer

This section describes single-rate color blind and color aware policers.

You can configure a single-rate color blind policer as follows:

```
[edit firewall three-color-policer foo]
single-rate {
    color-blind;
    committed-information-rate 50m;
    committed-burst-size 1500;
    excess-burst-size 1500;
}
```

You can apply the single-rate color blind policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-three-color foo;
```

You can configure a single-rate color-aware policer as follows:

```
[edit firewall three-color-policer bar]
single-rate {
    color-aware;
    committed-information-rate 50m;
    committed-burst-size 1500;
    excess-burst-size 1500;
}
```

You can apply the single-rate color-aware policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-three-color foo;
```

You also have the option to apply the policer at the physical port level as follows:

```
[edit interfaces so-0/1/0 layer2-policer]
input-three-color bar;
```

Configuring a Two-Rate Tricolor Marker Policer

Ingress policing is implemented using a two-rate tricolor marker (trTCM). This is done with a dual token bucket (DTB) that maintains two rates, committed, and a peak. Egress static policing also uses a token bucket.

The token buckets perform the following ingress policing functions:

- (1K) trTCM - Dual token bucket (red, yellow, and green marking)
- Policing is based on Layer 2 packet size:
  - After +/- byte adjust offset
- Marking is color aware and color blind:
  - Color aware needs to have the color set by q-tree lookup based on:
    - ToS
    - EXP
- Programmable marking actions:
  - Color (red, yellow, green)
  - Drop based on color and congestion profile
- Policer is selected based on the arriving channel number:
  - Channel number LUT produces policer index and queue index
  - Multiple channels can share the same policer (LUT produces same policer index)
- Support ingress policing and trTCM at the following levels:
  - Queue
  - Logical interface (ifl/DLCI)
  - Physical interface (ifd)
  - Physical port (controller ifd)
  - Any combinations of logical interface, physical interface, and port
- Support percentage of interface speed and bits per second

Rate limits may be applied to selected queues on ingress and on predefined queues at egress. The token bucket operates in color aware and color blind modes (specified by RFC 2698).

Configuring a Color-Blind trTCM

```
[edit firewall three-color-policer foo]
two-rate {
    color-blind;
    committed-information-rate 50m;
    committed-burst-size 1500;
    peak-information-rate 100m;
    peak-burst-size 3k;
}
```

You can apply the three-color two-rate color-blind policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-three-color foo;
```

You also have the option to apply the policer at the physical port level as follows:

```
[edit interfaces so-0/1/0 layer2-policer]
input-three-color foo;
```

Configuring a Color-Aware trTCM

```
[edit firewall three-color-policer bar]
two-rate {
    color-aware;
    committed-information-rate 50m;
    committed-burst-size 1500;
    peak-information-rate 100m;
    peak-burst-size 3k;
}
```

You can apply the three-color two-rate color-aware policer as follows:

```
[edit interfaces so-0/1/0 unit 0 layer2-policer]
input-three-color bar;
```

You also have the option to apply the policer at the physical port level as follows:

```
[edit interfaces so-0/1/0 layer2-policer]
input-three-color bar;
```

SEE ALSO

Class of Service User Guide (Routers and EX9200 Switches)

Three-Color Policer Configuration Guidelines

IN THIS SECTION

- Platforms Supported for Three-Color Policers | **2134**
- Color Modes for Three-Color Policers | **2135**
- Naming Conventions for Three-Color Policers | **2136**

Platforms Supported for Three-Color Policers

Three-color policers are supported on the following Juniper Networks routers:

- M120 Multiservice Edge Routers
- M320 Multiservice Edge Routers and T Series Core Routers with Enhanced II Flexible PIC Concentrators (FPCs)
- MX Series 5G Universal Routing Platforms
- T640 Core Routers with Enhanced Scaling FPC4
- T4000 Core Routers with FPC5

On MX Series and M120 routers, you can apply three-color policers to aggregated interfaces.

The `discard` action for a tricolor marking policer for a *firewall filter* is supported on the M120 routers, M320 routers with Enhanced-III FPCs, M7i and M10i routers with the Enhanced CFEB (CFEB-E), and

MX Series routers with Trio MPCs, so it is not necessary to include the `logical-interface-policer` statement for them.

SEE ALSO

Three-Color Policer Configuration Overview | 2118

Color Modes for Three-Color Policers

Naming Conventions for Three-Color Policers

Color Modes for Three-Color Policers

IN THIS SECTION

- Color-Blind Mode | 2135
- Color-Aware Mode | 2135

Three-color policers—both single-rate and two-rate three-color policer schemes—can operate in either of two modes:

Color-Blind Mode

In *color-blind* mode, the three-color policer assumes that all packets examined have not been previously marked or metered. If you configure a three-color policer to be color-blind instead of color-aware, the policer ignores preexisting color markings that might have been set for a packet by another traffic policer configured at a previous network node.

Color-Aware Mode

In *color-aware* mode, the three-color policer assumes that all packets examined have been previously marked or metered. In other words, the three-color policer takes into account any coloring markings that might have been set for a packet by another traffic policer configured at a previous network node. At the node where color-aware policing is configured, any preexisting color markings are used in determining the appropriate policing action for the packet.

In color-aware mode, the three-color policer can increase the packet loss priority (PLP) level of a packet, but never decrease it. For example, if a color-aware three-color policer meters a packet with a medium PLP marking, it can raise the PLP level to high, but cannot reduce the PLP level to low.

For two-rate, three-color policing, the Junos OS uses two token buckets to manage bandwidth based on the two rates of traffic. For example, two-rate policing might be configured on a node upstream in the network. The two-rate policer has marked a packet as yellow (loss priority medium-low). The color-aware policer takes this yellow marking into account when determining the appropriate policing action. In color-aware policing, the yellow packet would never receive the action associated with either the green packets or red packets. This way, tokens for violating packets are never taken from the metering token buckets at the color-aware policing node.

NOTE: For a three-color policer operating in color-aware mode and when the PLP of the input packet is medium-low, the color of the input packet to the policer is mapped to the color yellow. In such a scenario, if the color of the input packet remains unchanged, the policer operates in the following way:

- On a T1600 Enhanced Scaling Type 4 FPC (T1600-FPC4-ES), the PLP of the output packet remains medium-low.
- On a T4000 Type 5 FPC (T4000-FPC5-3D), the PLP of the output packet is marked as medium-high.

Because of this difference, for any applications (such as rewrite and WRED selection on egress interface) that use PLP, the packets are treated differently for the same flow depending on the FPC type (T1600 Enhanced Scaling FPC4 (T1600-FPC4-ES) or T4000 FPC5 (T4000-FPC5-3D)) on which the policer is applied.

SEE ALSO

Three-Color Policer Configuration Overview | 2118

Platforms Supported for Three-Color Policers

Naming Conventions for Three-Color Policers

Naming Conventions for Three-Color Policers

Because policers can be numerous and must be applied correctly to work, a simple naming convention makes it easier to apply the policers properly.

We recommend that you name your policer using a convention that identifies the basic components of the policer:

- Three-color policer type—Where `srTCM` identifies a *single-rate* three-color policer and `trTCM` identifies a *two-rate* three-color policer.

- Three-color policer color mode—Where `ca` identifies a *color-aware* three-color policer and `cb` identifies a *color-blind three-color policer*.

NOTE:
TCM stands for tricolor marking.

Table 119 on page 2137 describes a recommended naming convention for policers.

Table 119: Recommended Naming Convention for Policers

Three-Color Policer Type	Naming Convention	Example Names
Single-rate three-color, color-aware	`srTCM`*number*`-ca`	`srTCM1-ca`, `srTCM2-ca`, `srTCM3-ca`, ...
Single-rate three-color, color-blind	`srTCM`*number*`-cb`	`srTCM1-cb`, `srTCM2-cb`, `srTCM3-cb`, ...
Two-rate three-color, color-aware	`trTCM`*number*`-ca`	`trTCM1-ca`, `trTCM2-ca`, `trTCM3-ca`, ...
Two-rate three-color, color-blind	`trTCM`*number*`-cb`	`trTCM1-cb`, `trTCM2-cb`, `trTCM3-cb`, ...

SEE ALSO

Three-Color Policer Configuration Overview | 2118

Platforms Supported for Three-Color Policers

Color Modes for Three-Color Policers

RELATED DOCUMENTATION

Three-Color Policer Configuration Overview | 2118

Guidelines for Applying Traffic Policers | 1934

Basic Single-Rate Three-Color Policers

IN THIS SECTION

- Single-Rate Three-Color Policer Overview | 2138
- Example: Configuring a Single-Rate Three-Color Policer | 2139

Single-Rate Three-Color Policer Overview

A single-rate three-color policer defines a bandwidth limit and a maximum burst size for guaranteed traffic and a second burst size for peak traffic. A single-rate three-color policer is most useful when a service is structured according to packet length and not peak arrival rate.

Single-rate three-color policing meters a traffic stream based on the following configured traffic criteria:

- Committed information rate (CIR)—Bandwidth limit for guaranteed traffic.
- Committed burst size (CBS)—Maximum packet size permitted for bursts of data that exceed the CIR.
- Excess burst size (EBS)—Maximum packet size permitted for peak traffic.

Single-rate tricolor marking (single-rate TCM) classifies traffic as belonging to one of three color categories and performs congestion-control actions on the packets based on the color marking:

- Green—Traffic that conforms to *either* the bandwidth limit *or* the burst size for guaranteed traffic (CIR or CBS). For a green traffic flow, single-rate marks the packets with an implicit loss priority of `low` and transmits the packets.
- Yellow—Traffic that exceeds *both* the bandwidth limit *and* the burst size for guaranteed traffic (CIR and CBS) but not the burst size for peak traffic (EBS). For a yellow traffic flow, single-rate marks the packets with an implicit loss priority of `medium-high` and transmits the packets.
- Red—Traffic that exceeds the burst size for peak traffic (EBS), single-rate marks packets with an implicit loss priority of `high` and, optionally, discards the packets.

If congestion occurs downstream, the packets with higher loss priority are more likely to be discarded.

NOTE: For both single-rate and two-rate three-color policers, the only *configurable* action is to discard packets in a red traffic flow.

The `discard` action for a tricolor marking policer for a *firewall filter* is supported on the M120 routers, M320 routers with Enhanced-III FPCs, M7i and M10i routers with the Enhanced CFEB (CFEB-E), and MX Series routers with MPCs, so it is not necessary to include the `logical-interface-policer` statement for them.

SEE ALSO

Three-Color Policer Configuration Overview | 2118

Example: Configuring a Single-Rate Three-Color Policer

Example: Configuring a Single-Rate Three-Color Policer

IN THIS SECTION

- Requirements | 2139
- Overview | 2139
- Configuration | 2140
- Verification | 2146

This example shows how to configure a single-rate three-color policer.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 2140

A single-rate three-color policer meters a traffic flow against a bandwidth limit and burst-size limit for guaranteed traffic, plus a second burst-size limit for excess traffic. Traffic that conforms to the limits for guaranteed traffic is categorized as green, and nonconforming traffic falls into one of two categories:

- Nonconforming traffic that does not exceed the burst size for excess traffic is categorized as yellow.
- Nonconforming traffic that exceeds the burst size for excess traffic is categorized as red.

Each category is associated with an action. For green traffic, packets are implicitly set with a loss-priority value of `low` and then transmitted. For yellow traffic, packets are implicitly set with a loss-priority value of `medium-high` and then transmitted. For red traffic, packets are implicitly set with a loss-priority value of `high` and then transmitted. If the policer configuration includes the optional `action` statement (`action loss-priority high then discard`), then packets in a red flow are discarded instead.

You can apply a three-color policer to Layer 3 traffic as a firewall filter policer only. You reference the policer from a stateless firewall filter term, and then you apply the filter to the input or output of a logical interface at the protocol level.

Topology

In this example, you apply a color-aware, single-rate three-color policer to the input IPv4 traffic at logical interface `ge-2/0/5.0`. The IPv4 firewall filter term that references the policer does not apply any packet-filtering. The filter is used only to apply the three-color policer to the interface.

You configure the policer to rate-limit traffic to a bandwidth limit of 40 Mbps and a burst-size limit of 100 KB for green traffic but also allow an excess burst-size limit of 200 KB for yellow traffic. Only nonconforming traffic that exceeds the peak burst-size limit is categorized as red. In this example, you configure the three-color policer action `loss-priority high then discard`, which overrides the implicit marking of red traffic to a `high` loss priority.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2141**
- Configuring a Single-Rate Three-Color Policer | **2141**
- Configuring an IPv4 Stateless Firewall Filter That References the Policer | **2143**
- Applying the Filter to the Logical Interface | **2144**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set firewall three-color-policer srTCM1-ca single-rate color-aware
set firewall three-color-policer srTCM1-ca single-rate committed-information-rate 40m
set firewall three-color-policer srTCM1-ca single-rate committed-burst-size 100k
set firewall three-color-policer srTCM1-ca single-rate excess-burst-size 200k
set firewall three-color-policer srTCM1-ca action loss-priority high then discard
set firewall family inet filter filter-srtcm1ca-all term 1 then three-color-policer single-rate
srTCM1-ca
set class-of-service interfaces ge-2/0/5 unit 0 forwarding-class af
set interfaces ge-2/0/5 unit 0 family inet address 10.20.130.1/24
set interfaces ge-2/0/5 unit 0 family inet filter input filter-srtcm1ca-all
```

Configuring a Single-Rate Three-Color Policer

Step-by-Step Procedure

To configure a single-rate three-color policer:

1. Enable configuration of a three-color policer.

```
[edit]
user@host# edit firewall three-color-policer srTCM1-ca
```

2. Configure the color mode of the single-rate three-color policer.

```
[edit firewall three-color-policer srTCM1-ca]
user@host# set single-rate color-aware
```

3. Configure the single-rate guaranteed traffic limits.

```
[edit firewall three-color-policer srTCM1-ca]
user@host# set single-rate committed-information-rate 40m
user@host# set single-rate committed-burst-size 100k
```

4. Configure the single-rate burst-size limit that is used to classify nonconforming traffic.

```
[edit firewall three-color-policer srTCM1-ca]
user@host# set single-rate excess-burst-size 200k
```

5. (Optional) Configure the action for nonconforming traffic.

```
[edit firewall three-color-policer srTCM1-ca]
user@host# set action loss-priority high then discard
```

For three-color policers, the only configurable action is to discard packets in a red traffic flow. In this example, packets in a red traffic flow have been implicitly marked with a `high` packet loss priority (PLP) level because the traffic flow exceeded the rate-limiting defined by the single rate-limit (specified by the `committed-information-rate 40m` statement) and the larger burst-size limit (specified by the `excess-burst-size 200k` statement). Because the optional `action` statement is included, this example takes the more severe action of discarding packets in a red traffic flow.

Results

Confirm the configuration of the hierarchical policer by entering the `show firewall` configuration command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
three-color-policer srTCM1-ca {
    action {
        loss-priority high then discard;
    }
    single-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        excess-burst-size 200k;
```

```
    }
}
```

Configuring an IPv4 Stateless Firewall Filter That References the Policer

Step-by-Step Procedure

To configure a standard stateless firewall filter that references the policer:

1. Enable configuration of an IPv4 standard stateless firewall filter.

```
[edit]
user@host# edit firewall family inet filter filter-srtcm1ca-all
```

2. Specify the filter term that references the policer.

```
[edit firewall family inet filter filter-srtcm1ca-all]
user@host# set term 1 then three-color-policer single-rate srTCM1-ca
```

   Note that the term does not specify any match conditions. The firewall filter passes all packets to the policer.

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter-srtcm1ca-all {
        term 1 {
            then {
                three-color-policer {
                    single-rate srTCM1-ca;
                }
            }
        }
    }
```

```
}
three-color-policer srTCM1-ca {
    action {
        loss-priority high then discard;
    }
    single-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        excess-burst-size 200k;
    }
}
```

Applying the Filter to the Logical Interface

Step-by-Step Procedure

To apply the filter to the logical interface:

1. (MX Series routers only) (Optional) Reclassify all incoming packets on the logical interface `ge-2/0/5.0` to assured forwarding, regardless of any preexisting classification.

   ```
   [edit]
   user@host# set class-of-service interfaces ge-2/0/5 unit 0 forwarding-class af
   ```

   The classifier name can be a configured classifier or one of the default classifiers.

2. Enable configuration of the logical interface.

   ```
   [edit]
   user@host# edit interfaces ge-2/0/5 unit 0 family inet
   ```

3. Configure an IP address.

   ```
   [edit interfaces ge-2/0/5 unit 0 family inet]
   user@host# set address 10.20.130.1/24
   ```

4. Reference the filter as an input filter.

```
[edit interfaces ge-2/0/5 unit 0 family inet]
user@host# set filter input filter-srtcm1ca-all
```

Results

Confirm the configuration of the interface by entering the `show class-of-service` and `show interfaces` configuration mode commands. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show class-of-service
interfaces {
    ge-2/0/5 {
        unit 0 {
            forwarding-class af;
        }
    }
}

[edit]
user@host# show interfaces
ge-2/0/5 {
    unit 0 {
        family inet {
            filter {
                input filter-srtcm1ca-all;
                }
            address 10.20.130.1/24;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to the Logical Interface | **2146**

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to the Logical Interface

Purpose

Verify that the firewall filter is applied to IPv4 input traffic at the logical interface.

Action

Use the `show interfaces` operational mode command for the logical interface `ge-2/0/5.0`, and specify `detail` mode. The **Protocol inet** section of the command output displays IPv4 information for the logical interface. Within that section, the **Input Filters** field displays the name of the firewall filter applied to IPv4 input traffic at the logical interface.

```
user@host> show interfaces ge-2/0/5.0 detail
  Logical interface ge-2/0/5.0 (Index 105) (SNMP ifIndex 556) (Generation 170)
    Flags: Device-Down SNMP-Traps 0x4004000 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                   0 bps
     Output bytes  :                   0                   0 bps
     Input  packets:                   0                   0 pps
```

```
 Output packets:                    0                    0 pps
Protocol inet, MTU: 1500, Generation: 242, Route table: 0
  Flags: Sendbcast-pkt-to-re
  Input Filters: filter-srtcm1ca-all
  Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
    Destination: 10.20.130/24, Local: 10.20.130.1, Broadcast: 10.20.130.255,
    Generation: 171
Protocol multiservice, MTU: Unlimited, Generation: 243, Route table: 0
  Policer: Input: __default_arp_policer__
```

SEE ALSO

- Three-Color Policer Configuration Overview | 2118
- Single-Rate Three-Color Policer Overview

RELATED DOCUMENTATION

- Three-Color Policer Configuration Overview | 2118
- Three-Color Policer Configuration Guidelines | 2134

Basic Two-Rate Three-Color Policers

IN THIS SECTION

- Two-Rate Three-Color Policer Overview | 2147
- Example: Configuring a Two-Rate Three-Color Policer | 2149

Two-Rate Three-Color Policer Overview

A two-rate three-color policer defines two bandwidth limits (one for guaranteed traffic and one for peak traffic) and two burst sizes (one for each of the bandwidth limits). A two-rate three-color policer is most useful when a service is structured according to arrival rates and not necessarily packet length.

Two-rate three-color policing meters a traffic stream based on the following configured traffic criteria:

- Committed information rate (CIR)—Bandwidth limit for guaranteed traffic.
- Committed burst size (CBS)—Maximum packet size permitted for bursts of data that exceed the CIR.
- Peak information rate (PIR)—Bandwidth limit for peak traffic.
- Peak burst size (PBS)—Maximum packet size permitted for bursts of data that exceed the PIR.

Two-rate tricolor marking (two-rate TCM) classifies traffic as belonging to one of three color categories and performs congestion-control actions on the packets based on the color marking:

- Green—Traffic that conforms to the bandwidth limit and burst size for guaranteed traffic (CIR and CBS). For a green traffic flow, two-rate TCM marks the packets with an implicit loss priority of `low` and transmits the packets.
- Yellow—Traffic that exceeds the bandwidth limit or burst size for guaranteed traffic (CIR or CBS) but not the bandwidth limit and burst size for peak traffic (PIR and PBS). For a yellow traffic flow, two-rate TCM marks packets with an implicit loss priority of `medium-high` and transmits the packets.
- Red—Traffic that exceeds the bandwidth limit and burst size for peak traffic (PIR and PBS). For a red traffic flow, two-rate TCM marks packets with an implicit loss priority of `high` and, optionally, discards the packets.

If congestion occurs downstream, the packets with higher loss priority are more likely to be discarded.

NOTE: For both single-rate and two-rate three-color policers, the only *configurable* action is to discard packets in a red traffic flow.

For a tricolor marking policer referenced by a *firewall filter* term, the `discard` policing action is supported on the following routing platforms:

- EX Series switches
- M7i and M10i routers with the Enhanced CFEB (CFEB-E)
- M120 and M320 routers with Enhanced-III FPCs
- MX Series routers with Trio MPCs

To apply a tricolor marking policer on these routing platforms, it is not necessary to include the `logical-interface-policer` statement.

SEE ALSO

Example: Configuring a Two-Rate Three-Color Policer | 2157

Example: Configuring a Two-Rate Three-Color Policer

IN THIS SECTION

- Requirements | 2149
- Overview | 2149
- Configuration | 2150
- Verification | 2155

This example shows how to configure a two-rate three-color policer.

Requirements

Support for two-rate three-color policers varies according to the device. It includes SRX1400, SRX3400, SRX3600, SRX5400, SRX5600, and SRX5800 Firewall devices running a compatible version of Junos OS.

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 2150

A two-rate three-color policer meters a traffic flow against a bandwidth limit and burst-size limit for guaranteed traffic, plus a bandwidth limit and burst-size limit for peak traffic. Traffic that conforms to the limits for guaranteed traffic is categorized as green, and nonconforming traffic falls into one of two categories:

- Nonconforming traffic that does not exceed peak traffic limits is categorized as yellow.
- Nonconforming traffic that exceeds peak traffic limits is categorized as red.

Each category is associated with an action. For green traffic, packets are implicitly set with a loss-priority value of `low` and then transmitted. For yellow traffic, packets are implicitly set with a loss-priority value of `medium-high` and then transmitted. For red traffic, packets are implicitly set with a loss-priority value of `high`

and then transmitted. If the policer configuration includes the optional `action` statement (`action loss-priority high then discard`), then packets in a red flow are discarded instead.

You can apply a three-color policer to Layer 3 traffic as a firewall filter policer only. You reference the policer from a stateless firewall filter term, and then you apply the filter to the input or output of a logical interface at the protocol level.

Topology

In this example, you apply a color-aware, two-rate three-color policer to the input IPv4 traffic at logical interface `fe-0/1/1.0`. The IPv4 firewall filter term that references the policer does not apply any packet-filtering. The filter is used only to apply the three-color policer to the interface.

You configure the policer to rate-limit traffic to a bandwidth limit of 40 Mbps and a burst-size limit of 100 KB for green traffic, and you configure the policer to also allow a peak bandwidth limit of 60 Mbps and a peak burst-size limit of 200 KB for yellow traffic. Only nonconforming traffic that exceeds the peak traffic limits is categorized as red. In this example, you configure the three-color policer action `loss-priority high then discard`, which overrides the implicit marking of red traffic to a `high` loss priority.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2151**
- Configuring a Two-Rate Three-Color Policer | **2151**
- Configuring an IPv4 Stateless Firewall Filter That References the Policer | **2153**
- Applying the Filter to a Logical Interface at the Protocol Family Level | **2154**

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

To configure this example, perform the following tasks:

CLI Quick Configuration

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

```
set firewall three-color-policer trTCM1-ca two-rate color-aware
set firewall three-color-policer trTCM1-ca two-rate committed-information-rate 40m
set firewall three-color-policer trTCM1-ca two-rate committed-burst-size 100k
set firewall three-color-policer trTCM1-ca two-rate peak-information-rate 60m
set firewall three-color-policer trTCM1-ca two-rate peak-burst-size 200k
set firewall three-color-policer trTCM1-ca action loss-priority high then discard
set firewall family inet filter filter-trtcm1ca-all term 1 then three-color-policer two-rate
trTCM1-ca
set interfaces ge-2/0/5 unit 0 family inet address 10.10.10.1/30
set interfaces ge-2/0/5 unit 0 family inet filter input filter-trtcm1ca-all
set class-of-service interfaces ge-2/0/5 forwarding-class af
```

Configuring a Two-Rate Three-Color Policer

Step-by-Step Procedure

To configure a two-rate three-color policer:

1. Enable configuration of a three-color policer.

```
[edit]
user@host# set firewall three-color-policer trTCM1-ca
```

2. Configure the color mode of the two-rate three-color policer.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate color-aware
```

3. Configure the two-rate guaranteed traffic limits.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate committed-information-rate 40m
user@host# set two-rate committed-burst-size 100k
```

Traffic that does not exceed both of these limits is categorized as green. Packets in a green flow are implicitly set to `low` loss priority and then transmitted.

4. Configure the two-rate peak traffic limits.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate peak-information-rate 60m
user@host# set two-rate peak-burst-size 200k
```

Nonconforming traffic that does not exceed both of these limits is categorized as yellow. Packets in a yellow flow are implicitly set to `medium-high` loss priority and then transmitted. Nonconforming traffic that exceeds both of these limits is categorized as red. Packets in a red flow are implicitly set to `high` loss priority.

5. (Optional) Configure the policer action for red traffic.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set action loss-priority high then discard
```

For three-color policers, the only configurable action is to discard red packets. Red packets are packets that have been assigned high loss priority because they exceeded the peak information rate (PIR) and the peak burst size (PBS).

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
three-color-policer trTCM1-ca {
    action {
        loss-priority high then discard;
```

```
    }
    two-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Configuring an IPv4 Stateless Firewall Filter That References the Policer

Step-by-Step Procedure

To configure an IPv4 stateless firewall filter that references the policer:

1. Enable configuration of an IPv4 standard stateless firewall filter.

```
[edit]
user@host# set firewall family inet filter filter-trtcm1ca-all
```

2. Specify the filter term that references the policer.

```
[edit firewall family inet filter filter-trtcm1ca-all]
user@host# set term 1 then three-color-policer two-rate trTCM1-ca
```

   Note that the term does not specify any match conditions. The firewall filter passes all packets to the policer.

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter-trtcm1ca-all {
        term 1 {
```

```
            then {
                three-color-policer {
                    two-rate trTCM1-ca;
                }
            }
        }
    }
}
three-color-policer trTCM1-ca {
    action {
        loss-priority high then discard;
    }
    two-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Applying the Filter to a Logical Interface at the Protocol Family Level

Step-by-Step Procedure

To apply the filter to the logical interface at the protocol family level:

1. Enable configuration of an IPv4 firewall filter.

```
[edit]
user@host# edit interfaces ge-2/0/5 unit 0 family inet
```

2. Apply the policer to the logical interface at the protocol family level.

```
[edit interfaces ge-2/0/5 unit 0 family inet]
user@host# set address 10.10.10.1/30
user@host# set filter input filter-trtcm1ca-all
```

3. (MX Series routers and EX Series switches only) (Optional) For input policers, you can configure a fixed classifier. A fixed classifier reclassifies all incoming packets, regardless of any preexisting classification.

   NOTE: Platform support depends on the Junos OS release in your implementation.

   ```
   [edit]
   user@host# set class-of-service interfaces ge-2/0/5 forwarding-class af
   ```

   The classifier name can be a configured classifier or one of the default classifiers.

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-2/0/5 {
    unit 0 {
        family inet {
            address 10.10.10.1/30;
            filter {
                input filter-trtcm1ca-all;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to the Logical Interface | **2156**

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to the Logical Interface

Purpose

Verify that the firewall filter is applied to IPv4 input traffic at the logical interface.

Action

Use the `show interfaces` operational mode command for the logical interface `ge-2/0/5.0`, and specify `detail` mode. The **Protocol inet** section of the command output displays IPv4 information for the logical interface. Within that section, the **Input Filters** field displays the name of IPv4 firewall filters associated with the logical interface.

```
user@host> show interfaces ge-2/0/5.0 detail
  Logical interface ge-2/0/5.0 (Index 105) (SNMP ifIndex 556) (Generation 170)
    Flags: Device-Down SNMP-Traps 0x4004000 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                   0 bps
     Output bytes  :                   0                   0 bps
     Input  packets:                   0                   0 pps
     Output packets:                   0                   0 pps
    Protocol inet, MTU: 1500, Generation: 242, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter-trtcm1ca-all
      Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
        Destination: 10.20.130/24, Local: 10.20.130.1, Broadcast: 10.20.130.255,
        Generation: 171
    Protocol multiservice, MTU: Unlimited, Generation: 243, Route table: 0
      Policer: Input: __default_arp_policer__
```

SEE ALSO

Two-Rate Three-Color Policer Overview | 2147

RELATED DOCUMENTATION

Three-Color Policer Configuration Overview | 2118

Three-Color Policer Configuration Guidelines | 2134

Example: Configuring a Two-Rate Three-Color Policer

IN THIS SECTION

- Requirements | 2157
- Overview | 2157
- Configuration | 2158
- Verification | 2163

This example shows how to configure a two-rate three-color policer.

Requirements

Support for two-rate three-color policers varies according to the device. It includes SRX1400, SRX3400, SRX3600, SRX5400, SRX5600, and SRX5800 Firewall devices running a compatible version of Junos OS.

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 2158

A two-rate three-color policer meters a traffic flow against a bandwidth limit and burst-size limit for guaranteed traffic, plus a bandwidth limit and burst-size limit for peak traffic. Traffic that conforms to the limits for guaranteed traffic is categorized as green, and nonconforming traffic falls into one of two categories:

- Nonconforming traffic that does not exceed peak traffic limits is categorized as yellow.
- Nonconforming traffic that exceeds peak traffic limits is categorized as red.

Each category is associated with an action. For green traffic, packets are implicitly set with a loss-priority value of `low` and then transmitted. For yellow traffic, packets are implicitly set with a loss-priority value of `medium-high` and then transmitted. For red traffic, packets are implicitly set with a loss-priority value of `high` and then transmitted. If the policer configuration includes the optional `action` statement (`action loss-priority high then discard`), then packets in a red flow are discarded instead.

You can apply a three-color policer to Layer 3 traffic as a firewall filter policer only. You reference the policer from a stateless firewall filter term, and then you apply the filter to the input or output of a logical interface at the protocol level.

Topology

In this example, you apply a color-aware, two-rate three-color policer to the input IPv4 traffic at logical interface `fe-0/1/1.0`. The IPv4 firewall filter term that references the policer does not apply any packet-filtering. The filter is used only to apply the three-color policer to the interface.

You configure the policer to rate-limit traffic to a bandwidth limit of 40 Mbps and a burst-size limit of 100 KB for green traffic, and you configure the policer to also allow a peak bandwidth limit of 60 Mbps and a peak burst-size limit of 200 KB for yellow traffic. Only nonconforming traffic that exceeds the peak traffic limits is categorized as red. In this example, you configure the three-color policer action `loss-priority high then discard`, which overrides the implicit marking of red traffic to a `high` loss priority.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2159**
- Configuring a Two-Rate Three-Color Policer | **2159**
- Configuring an IPv4 Stateless Firewall Filter That References the Policer | **2161**
- Applying the Filter to a Logical Interface at the Protocol Family Level | **2162**

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

To configure this example, perform the following tasks:

CLI Quick Configuration

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

```
set firewall three-color-policer trTCM1-ca two-rate color-aware
set firewall three-color-policer trTCM1-ca two-rate committed-information-rate 40m
set firewall three-color-policer trTCM1-ca two-rate committed-burst-size 100k
set firewall three-color-policer trTCM1-ca two-rate peak-information-rate 60m
set firewall three-color-policer trTCM1-ca two-rate peak-burst-size 200k
set firewall three-color-policer trTCM1-ca action loss-priority high then discard
set firewall family inet filter filter-trtcm1ca-all term 1 then three-color-policer two-rate
trTCM1-ca
set interfaces ge-2/0/5 unit 0 family inet address 10.10.10.1/30
set interfaces ge-2/0/5 unit 0 family inet filter input filter-trtcm1ca-all
set class-of-service interfaces ge-2/0/5 forwarding-class af
```

Configuring a Two-Rate Three-Color Policer

Step-by-Step Procedure

To configure a two-rate three-color policer:

1. Enable configuration of a three-color policer.

```
[edit]
user@host# set firewall three-color-policer trTCM1-ca
```

2. Configure the color mode of the two-rate three-color policer.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate color-aware
```

3. Configure the two-rate guaranteed traffic limits.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate committed-information-rate 40m
user@host# set two-rate committed-burst-size 100k
```

Traffic that does not exceed both of these limits is categorized as green. Packets in a green flow are implicitly set to `low` loss priority and then transmitted.

4. Configure the two-rate peak traffic limits.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set two-rate peak-information-rate 60m
user@host# set two-rate peak-burst-size 200k
```

Nonconforming traffic that does not exceed both of these limits is categorized as yellow. Packets in a yellow flow are implicitly set to `medium-high` loss priority and then transmitted. Nonconforming traffic that exceeds both of these limits is categorized as red. Packets in a red flow are implicitly set to `high` loss priority.

5. (Optional) Configure the policer action for red traffic.

```
[edit firewall three-color-policer trTCM1-ca]
user@host# set action loss-priority high then discard
```

For three-color policers, the only configurable action is to discard red packets. Red packets are packets that have been assigned high loss priority because they exceeded the peak information rate (PIR) and the peak burst size (PBS).

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
three-color-policer trTCM1-ca {
    action {
        loss-priority high then discard;
```

```
    }
    two-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Configuring an IPv4 Stateless Firewall Filter That References the Policer

Step-by-Step Procedure

To configure an IPv4 stateless firewall filter that references the policer:

1. Enable configuration of an IPv4 standard stateless firewall filter.

```
[edit]
user@host# set firewall family inet filter filter-trtcm1ca-all
```

2. Specify the filter term that references the policer.

```
[edit firewall family inet filter filter-trtcm1ca-all]
user@host# set term 1 then three-color-policer two-rate trTCM1-ca
```

Note that the term does not specify any match conditions. The firewall filter passes all packets to the policer.

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter-trtcm1ca-all {
        term 1 {
```

```
            then {
                three-color-policer {
                    two-rate trTCM1-ca;
                }
            }
        }
    }
}
three-color-policer trTCM1-ca {
    action {
        loss-priority high then discard;
    }
    two-rate {
        color-aware;
        committed-information-rate 40m;
        committed-burst-size 100k;
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Applying the Filter to a Logical Interface at the Protocol Family Level

Step-by-Step Procedure

To apply the filter to the logical interface at the protocol family level:

1. Enable configuration of an IPv4 firewall filter.

```
[edit]
user@host# edit interfaces ge-2/0/5 unit 0 family inet
```

2. Apply the policer to the logical interface at the protocol family level.

```
[edit interfaces ge-2/0/5 unit 0 family inet]
user@host# set address 10.10.10.1/30
user@host# set filter input filter-trtcm1ca-all
```

3. (MX Series routers and EX Series switches only) (Optional) For input policers, you can configure a fixed classifier. A fixed classifier reclassifies all incoming packets, regardless of any preexisting classification.

   NOTE: Platform support depends on the Junos OS release in your implementation.

   ```
   [edit]
   user@host# set class-of-service interfaces ge-2/0/5 forwarding-class af
   ```

   The classifier name can be a configured classifier or one of the default classifiers.

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-2/0/5 {
    unit 0 {
        family inet {
            address 10.10.10.1/30;
            filter {
                input filter-trtcm1ca-all;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to the Logical Interface | **2164**

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to the Logical Interface

Purpose

Verify that the firewall filter is applied to IPv4 input traffic at the logical interface.

Action

Use the `show interfaces` operational mode command for the logical interface `ge-2/0/5.0`, and specify `detail` mode. The **Protocol inet** section of the command output displays IPv4 information for the logical interface. Within that section, the **Input Filters** field displays the name of IPv4 firewall filters associated with the logical interface.

```
user@host> show interfaces ge-2/0/5.0 detail
  Logical interface ge-2/0/5.0 (Index 105) (SNMP ifIndex 556) (Generation 170)
    Flags: Device-Down SNMP-Traps 0x4004000 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                   0 bps
     Output bytes  :                   0                   0 bps
     Input  packets:                   0                   0 pps
     Output packets:                   0                   0 pps
    Protocol inet, MTU: 1500, Generation: 242, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter-trtcm1ca-all
      Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
        Destination: 10.20.130/24, Local: 10.20.130.1, Broadcast: 10.20.130.255,
        Generation: 171
    Protocol multiservice, MTU: Unlimited, Generation: 243, Route table: 0
      Policer: Input: __default_arp_policer__
```

RELATED DOCUMENTATION

Two-Rate Three-Color Policer Overview | 2147

CHAPTER 33

Configuring Logical and Physical Interface Traffic Policers at Layer 3

IN THIS CHAPTER

- Two-Color and Three-Color Logical Interface Policers | **2166**
- Two-Color and Three-Color Physical Interface Policers | **2185**

Two-Color and Three-Color Logical Interface Policers

IN THIS SECTION

- Logical Interface (Aggregate) Policer Overview | **2166**
- Example: Configuring a Two-Color Logical Interface (Aggregate) Policer | **2167**
- Example: Configuring a Three-Color Logical Interface (Aggregate) Policer | **2176**

Logical Interface (Aggregate) Policer Overview

A *logical interface policer*—also called an *aggregate policer*—is a two-color or three-color policer that defines traffic rate limiting that you can apply to input or output traffic for multiple protocol families on the same logical interface without creating multiple instances of the policer.

To configure a single-rate two-color *logical interface* policer, include the `logical-interface-policer` statement at one of the following hierarchy levels:

- [edit `firewall policer` *policer-name*]
- [edit logical-systems *logical-system-name* `firewall policer` *policer-name*]

To configure a single-rate or two-rate three-color logical interface policer, include the `logical-interface-policer` statement at one of the following hierarchy levels:

- [edit firewall three-color-policer *name*]
- [edit logical-systems *logical-system-name* firewall three-color-policer *name*]

NOTE: A three-color policer can be applied to Layer 2 traffic as a logical interface policer only. You cannot apply a three-color policer to Layer 2 traffic as a physical interface policer (through a *firewall filter*).

You apply a logical interface policer to Layer 3 traffic directly to the interface configuration at the logical unit level (to rate-limit all traffic types, regardless of the protocol family) or at the protocol family level (to rate-limit traffic of a specific protocol family). It is OK to reference a logical interface policer from a stateless firewall filter term and then apply the filter to a logical interface.

You can apply a logical interface policer to unicast traffic only. For information about configuring a stateless firewall filter for flooded traffic, see "*Applying Forwarding Table Filters*" in the "Traffic Sampling, Forwarding, and Monitoring" section of the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

To display a logical interface policer on a particular interface, issue the `show interfaces policers` *operational mode command*.

SEE ALSO

Two-Color Policer Configuration Overview | **2034**

Three-Color Policer Configuration Overview | **2118**

Example: Configuring a Two-Color Logical Interface (Aggregate) Policer

Example: Configuring a Three-Color Logical Interface (Aggregate) Policer

interface-specific (Firewall Filters)

logical-interface-policer

Example: Configuring a Two-Color Logical Interface (Aggregate) Policer

IN THIS SECTION

- Requirements | **2168**
- Overview | **2168**
- Configuration | **2169**

- Verification | 2174

This example shows how to configure a single-rate two-color policer as a logical interface policer and apply it to incoming IPv4 traffic on a logical interface.

Requirements

Before you begin, make sure that the logical interface to which you apply the two-color logical interface policer is hosted on a Gigabit Ethernet interface (`ge-`) or a 10-Gigabit Ethernet interface (`xe-`).

Overview

IN THIS SECTION

- Topology | 2168

In this example, you configure the single-rate two-color policer `policer_IFL` as a logical interface policer and apply it to incoming IPv4 traffic at logical interface `ge-1/3/1.0`.

Topology

If the input IPv4 traffic on the physical interface `ge-1/3/1` exceeds the bandwidth limit equal to 90 percent of the media rate with a 300 KB burst-size limit, then the logical interface policer `policer_IFL` rate-limits the input IPv4 traffic on the logical interface `ge-1/3/1.0`. Configure the policer to mark nonconforming traffic by setting packet loss priority (PLP) levels to `high` and classifying packets as `best-effort`.

As the incoming IPv4 traffic rate on the physical interface slows and conforms to the configured limits, Junos OS stops marking the incoming IPv4 packets at the logical interface.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 2169
- Configuring the Logical Interfaces | 2169
- Configuring the Single-Rate Two-Color Policer as a Logical Interface Policer | 2171
- Applying the Logical Interface Policer to Input IPv4 Traffic at a Logical Interface | 2173

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces ge-1/3/1 vlan-tagging
set interfaces ge-1/3/1 unit 0 vlan-id 100
set interfaces ge-1/3/1 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/3/1 unit 1 vlan-id 101
set interfaces ge-1/3/1 unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac
00:00:11:22:33:44
set firewall policer policer_IFL logical-interface-policer
set firewall policer policer_IFL if-exceeding bandwidth-percent 90
set firewall policer policer_IFL if-exceeding burst-size-limit 300k
set firewall policer policer_IFL then loss-priority high
set firewall policer policer_IFL then forwarding-class best-effort
set interfaces ge-1/3/1 unit 0 family inet policer input policer_IFL
```

Configuring the Logical Interfaces

Step-by-Step Procedure

To configure the logical interfaces:

1. Enable configuration of the interface.

```
[edit]
user@host# edit interfaces ge-1/3/1
```

2. Configure single tagging.

```
[edit interfaces ge-1/3/1]
user@host# set vlan-tagging
```

3. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 0 vlan-id 100
user@host# set unit 0 family inet address 10.10.10.1/30
```

4. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 1 vlan-id 101
user@host# set unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac 00:00:11:22:33:44
```

Results

Confirm the configuration of the logical interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.10.10.1/30;
        }
```

```
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

Configuring the Single-Rate Two-Color Policer as a Logical Interface Policer

Step-by-Step Procedure

To configure a single-rate two-color policer as a logical interface policer:

1. Enable configuration of a single-rate two-color policer.

   ```
   [edit]
   user@host# edit firewall policer policer_IFL
   ```

2. Specify that the policer is a logical interface (aggregate) policer.

   ```
   [edit firewall policer policer_IFL]
   user@host# set logical-interface-policer
   ```

   A logical interface policer rate-limits traffic based on a percentage of the media rate of the physical interface underlying the logical interface to which the policer is applied. The policer is applied directly to the interface rather than referenced by a firewall filter.

3. Specify the policer traffic limits.

   - Specify the bandwidth limit.
     - To specify the bandwidth limit as an absolute rate, from 8,000 bits per second through 50,000,000,000 bits per second, include the `bandwidth-limit` *`bps`* statement.
     - To specify the bandwidth limit as a percentage of the physical port speed on the interface, include the `bandwidth-percent` *`percent`* statement.

In this example, the CLI commands and output are based on a bandwidth limit specified as a percentage rather than as an absolute rate.

```
[edit firewall policer policer_IFL]
user@host# set if-exceeding bandwidth-percent 90
```

- Specify the burst-size limit, from 1,500 bytes through 100,000,000,000 bytes, which is the maximum packet size to be permitted for bursts of data that exceed the specified bandwidth limit.

```
[edit firewall policer policer_IFL]
user@host# set if-exceeding burst-size-limit 300k
```

4. Specify the policer actions to be taken on traffic that exceeds the configured rate limits.

   - To discard the packet, include the `discard` statement.
   - To set the loss-priority value of the packet, include the `loss-priority (low | medium-low | medium-high | high)` statement.
   - To classify the packet to a forwarding class, include the `forwarding-class (`*`forwarding-class`*` | assured-forwarding | best-effort | expedited-forwarding | network-control)` statement.

   In this example, the CLI commands and output are based on both setting the packet loss priority level and classifying the packet.

```
[edit firewall policer policer_IFL]
user@host# set then loss-priority high
user@host# set then forwarding-class best-effort
```

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer policer_IFL {
    logical-interface-policer;
    if-exceeding {
```

```
        bandwidth-percent 90;
        burst-size-limit 300k;
    }
    then {
        loss-priority high;
        forwarding-class best-effort;
    }
}
```

Applying the Logical Interface Policer to Input IPv4 Traffic at a Logical Interface

Step-by-Step Procedure

To apply the two-color logical interface policer to input IPv4 traffic a logical interface:

1. Enable configuration of the logical interface.

```
[edit]
user@host# edit interfaces ge-1/3/1 unit 0
```

2. Apply the policer to all traffic types or to a specific traffic type on the logical interface.

   - To apply the policer to all traffic types, regardless of the protocol family, include the `policer (input | output)` *policer-name* statement at the `[edit interfaces` *interface-name* `unit` *number*`]` hierarchy level.
   - To apply the policer to traffic of a specific protocol family, include the `policer (input | output)` *policer-name* statement at the `[edit interfaces` *interface-name* `unit` *unit-number* `family` *family-name*`]` hierarchy level.

   To apply the logical interface policer to incoming packets, use the `policer input` *policer-name* statement. To apply the logical interface policer to outgoing packets, use the `policer output` *policer-name* statement.

   In this example, the CLI commands and output are based on rate-limiting the IPv4 input traffic at logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1 unit 0]
user@host# set family inet policer input policer_IFL
```

Results

Confirm the configuration of the interface by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            policer input policer_IFL;
            address 10.10.10.1/30;
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | **2175**
- Displaying Statistics for the Policer | **2175**

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interface `ge-1/3/1.0`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface. The **Protocol inet** subsection contains a **Policer** field that would list the policer `policer_IFL` as an input or output logical interface policer as follows:

- **Input: policer_IFL-ge-1/3/1.0-log_int-i**
- **Output: policer_IFL-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to input traffic only.

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction. For the policer `policer_IFL`, the input and output policer names are displayed as follows:

- **policer_IFL-ge-1/3/1.0-log_int-i**
- **policer_IFL-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to input traffic only.

SEE ALSO

Two-Color Policer Configuration Overview | 2034

Logical Interface (Aggregate) Policer Overview

Example: Configuring a Three-Color Logical Interface (Aggregate) Policer

IN THIS SECTION

- Requirements | 2176
- Overview | 2176
- Configuration | 2177
- Verification | 2183

This example shows how to configure a two-rate three-color color-blind policer as a logical interface (aggregate) policer and apply the policer directly to Layer 2 input traffic at a supported logical interface.

Requirements

Before you begin, make sure that the logical interface to which you apply the three-color logical interface policer is hosted on a Gigabit Ethernet interface (`ge-`) or a 10-Gigabit Ethernet interface (`xe-`) on an MX Series router.

Overview

IN THIS SECTION

- Topology | 2177

A two-rate three-color policer meters a traffic flow against a bandwidth limit and burst-size limit for guaranteed traffic, plus a second set of bandwidth and burst-size limits for peak traffic. Traffic that conforms to the limits for guaranteed traffic is categorized as green, and nonconforming traffic falls into one of two categories:

- Nonconforming traffic that does not exceed the bandwidth and burst-size limits for peak traffic is categorized as yellow.

- Nonconforming traffic that exceeds the bandwidth and burst-size limits for peak traffic is categorized as red.

A logical interface policer defines traffic rate-limiting rules that you can apply to multiple protocol families on the same logical interface without creating multiple instances of the policer.

NOTE: You apply a logical interface policer directly to a logical interface at the logical unit level, and not by referencing the policer in a stateless firewall filter and then applying the filter to the logical interface at the protocol family level.

Topology

In this example, you configure the two-rate three-color policer `trTCM2-cb` as a color-blind logical interface policer and apply the policer to incoming Layer 2 traffic on logical interface `ge-1/3/1.0`.

NOTE: When using a three-color policer to rate-limit Layer 2 traffic, color-aware policing can be applied to egress traffic only.

The policer defines guaranteed traffic rate limits such that traffic that conforms to the bandwidth limit of 40 Mbps with a 100 KB allowance for traffic bursting (based on the token-bucket formula) is categorized as green. As with any policed traffic, the packets in a green flow are implicitly set to a `low` loss priority and then transmitted.

Nonconforming traffic that falls within the peak traffic limits of a 60 Mbps bandwidth limit and a 200 KB allowance for traffic bursting (based on the token-bucket formula) is categorized as yellow. The packets in a yellow traffic flow are implicitly set to a `medium-high` loss priority and then transmitted.

Nonconforming traffic that exceeds the peak traffic limits are categorized as red. The packets in a red traffic flow are implicitly set to a `high` loss priority. In this example, the optional policer action for red traffic (`loss-priority high then discard`) is configured, so packets in a red traffic flow are discarded instead of transmitted.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2178**
- Configuring the Logical Interfaces | **2178**

Configuring the Two-Rate Three-Color Policer as a Logical Interface Policer | 2180
Applying the Three-Color Policer to the Layer 2 Input at the Logical Interface | 2182

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces ge-1/3/1 vlan-tagging
set interfaces ge-1/3/1 unit 0 vlan-id 100
set interfaces ge-1/3/1 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/3/1 unit 1 vlan-id 101
set interfaces ge-1/3/1 unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac
00:00:11:22:33:44
set firewall three-color-policer trTCM2-cb logical-interface-policer
set firewall three-color-policer trTCM2-cb two-rate color-blind
set firewall three-color-policer trTCM2-cb two-rate committed-information-rate 40m
set firewall three-color-policer trTCM2-cb two-rate committed-burst-size 100k
set firewall three-color-policer trTCM2-cb two-rate peak-information-rate 60m
set firewall three-color-policer trTCM2-cb two-rate peak-burst-size 200k
set firewall three-color-policer trTCM2-cb action loss-priority high then discard
set interfaces ge-1/3/1 unit 0 layer2-policer input-three-color trTCM2-cb
```

Configuring the Logical Interfaces

Step-by-Step Procedure

To configure the logical interfaces:

1. Enable configuration of the interface.

```
[edit]
user@host# edit interfaces ge-1/3/1
```

2. Configure single tagging.

```
[edit interfaces ge-1/3/1]
user@host# set vlan-tagging
```

3. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 0 vlan-id 100
user@host# set unit 0 family inet address 10.10.10.1/30
```

4. Configure logical interface `ge-1/3/1.0`.

```
[edit interfaces ge-1/3/1]
user@host# set unit 1 vlan-id 101
user@host# set unit 1 family inet address 20.20.20.1/30 arp 20.20.20.2 mac 00:00:11:22:33:44
```

Results

Confirm the configuration of the logical interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        family inet {
            address 10.10.10.1/30;
        }
```

```
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

Configuring the Two-Rate Three-Color Policer as a Logical Interface Policer

Step-by-Step Procedure

To configure the two-rate three-color policer as a logical interface policer:

1. Enable configuration of a three-color policer.

```
[edit]
user@host# edit firewall three-color-policer trTCM2-cb
```

2. Specify that the policer is a logical interface (aggregate) policer.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set logical-interface-policer
```

A logical interface policer rate-limits traffic based on a percentage of the media rate of the physical interface underlying the logical interface to which the policer is applied, and the policer is applied directly to the interface rather than referenced by a firewall filter.

3. Specify that the policer is two-rate and color-blind.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate color-blind
```

A color-aware three-color policer takes into account any coloring markings that might have been set for a packet by another traffic policer configured at a previous network node, and any preexisting color markings are used in determining the appropriate policing action for the packet.

Because you are applying this three-color policer applied to input at Layer 2, you must configure the policer to be color-blind.

4. Specify the policer traffic limits used to classify a green traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate committed-information-rate 40m
user@host# set two-rate committed-burst-size 100k
```

5. Specify the additional policer traffic limits used to classify a yellow or red traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set two-rate peak-information-rate 60m
user@host# set two-rate peak-burst-size 200k
```

6. (Optional) Specify the configured policer action for packets in a red traffic flow.

```
[edit firewall three-color-policer trTCM2-cb]
user@host# set action loss-priority high then discard
```

In color-aware mode, the three-color policer configured action can increase the packet loss priority (PLP) level of a packet, but never decrease it. For example, if a color-aware three-color policer meters a packet with a medium PLP marking, it can raise the PLP level to high, but cannot reduce the PLP level to low.

Results

Confirm the configuration of the three-color policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
three-color-policer trTCM2-cb {
    logical-interface-policer;
    action {
        loss-priority high then discard;
    }
    two-rate {
```

```
        color-blind;
        committed-information-rate 40m;
        committed-burst-size 100k;
        peak-information-rate 60m;
        peak-burst-size 200k;
    }
}
```

Applying the Three-Color Policer to the Layer 2 Input at the Logical Interface

Step-by-Step Procedure

To apply the three-color policer to the Layer 2 input at the logical interface:

1. Enable application of Layer 2 logical interface policers.

```
[edit]
user@host# edit interfaces ge-1/3/1 unit 0
```

2. Apply the three-color logical interface policer to a logical interface input.

```
[edit interfaces ge-1/3/1 unit 0]
user@host# set layer2-policerinput-three-color trTCM2-cb
```

Results

Confirm the configuration of the logical interfaces by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 100;
        layer2-policer {
            input-three-color trTCM2-cb;
        }
```

```
        family inet {
            address 10.10.10.1/30;
        }
    }
    unit 1 {
        vlan-id 101;
        family inet {
            address 20.20.20.1/30 {
                arp 20.20.20.2 mac 00:00:11:22:33:44;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying Traffic Statistics and Policers for the Logical Interface | **2183**
- Displaying Statistics for the Policer | **2184**

Confirm that the configuration is working properly.

Displaying Traffic Statistics and Policers for the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show interfaces` operational mode command for logical interface `ge-1/3/1.0`, and include the `detail` or `extensive` option. The command output section for **Traffic statistics** lists the number of bytes and packets received and transmitted on the logical interface, and the **Protocol inet** section contains a **Policer** field that would list the policer `trTCM2-cb` as an input or output policer as follows:

- **Input: trTCM2-cb-ge-1/3/1.0-log_int-i**
- **Output: trTCM2-cb-ge-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to in the input direction only.

Displaying Statistics for the Policer

Purpose

Verify the number of packets evaluated by the policer.

Action

Use the `show policer` operational mode command and optionally specify the name of the policer. The command output displays the number of packets evaluated by each configured policer (or the specified policer), in each direction. For the policer `trTCM2-cb`, the input and output policer names are displayed as follows:

- **trTCM2-cb-ge-1/3/1.0-log_int-i**
- **trTCM2-cb-e-1/3/1.0-log_int-o**

The **log_int-i** suffix denotes a logical interface policer applied to input traffic, while the **log_int-o** suffix denotes a logical interface policer applied to output traffic. In this example, the logical interface policer is applied to input traffic only.

SEE ALSO

Logical Interface (Aggregate) Policer Overview

Example: Configuring a Two-Color Logical Interface (Aggregate) Policer

Three-Color Policing at Layer 2 Overview

layer2-policer

logical-interface-policer

three-color-policer (Configuring)

RELATED DOCUMENTATION

Two-Color Policer Configuration Overview | 2034

Three-Color Policer Configuration Overview | 2118

Guidelines for Applying Traffic Policers | 1934

Two-Color and Three-Color Physical Interface Policers

IN THIS SECTION

- Physical Interface Policer Overview | 2185
- Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface | 2187

Physical Interface Policer Overview

A *physical interface policer* is a two-color or three-color policer that defines traffic rate limiting that you can apply to input or output traffic for all the logical interfaces and protocol families configured on a physical interface, even if the logical interfaces belong to different routing 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, suppose that 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. Now suppose that a customer wants to apply one set of aggregated rate limits for certain types of traffic on a single physical interface. To accomplish this, you could apply a single physical interface policer to the physical interface, which rate-limits all the logical interfaces configured on the interface and all the routing instances to which those interfaces belong.

To configure a single-rate two-color physical interface policer, include the `physical-interface-policer` statement at one of the following hierarchy levels:

- [edit `firewall policer` *policer-name*]
- [edit logical-system *logical-system-name* `firewall policer` *policer-name*]
- [edit routing-instances *routing-instance-name* `firewall policer` *policer-name*]
- [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* `firewall policer` *policer-name*]

To configure a single-rate or two-rate three-color physical interface policer, include the `physical-interface-policer` statement at one of the following hierarchy levels:

- [edit firewall three-color-policer *policer-name*]
- [edit logical-system *logical-system-name* firewall three-color-policer *policer-name*]
- [edit routing-instances *routing-instance-name* firewall three-color-policer *policer-name*]
- [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* firewall three-color-policer *policer-name*]

You apply a physical interface policer to Layer 3 traffic by referencing the policer from a stateless *firewall filter* term and then applying the filter to a *logical interface*. You cannot apply a physical interface to Layer 3 traffic directly to the interface configuration.

To reference a single-rate two-color policer from a stateless firewall filter term, use the `policer` nonterminating action. To reference a single-rate or two-rate three-color policer from a stateless firewall filter term, use the `three-color-policer` nonterminating action.

The following requirements apply to a stateless firewall filter that references a physical interface policer:

- You must configure the firewall filter for a specific, supported protocol family: `ipv4`, `ipv6`, `mpls`, `vpls`, or circuit cross-connect (`ccc`), but not for `family any`.
- You must configure the firewall filter as a *physical interface filter* by including the `physical-interface-filter` statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level.
- A firewall filter that is defined as a physical interface filter can reference a physical interface policer only.
- A firewall filter that is defined on the global (non-logical) system cannot be used in a logical system for interface-specific filter instances. More specifically, you cannot use a template for a **physical-interface-filter** that was created on the global system with a filter attachment that was created on the logical system. Both the template and the attachment must reside on the logical system for filtering to work correctly. This is because, for logical systems, filter instance naming is derived from the physical interface, but the same is not true for interface-specific filter instances.
- A firewall filter that is defined as a physical interface filter cannot reference a policer configured with the `interface-specific` statement.
- You cannot configure a firewall filter as both a physical interface filter and as a logical interface filter that also includes the `interface-specific` statement.

SEE ALSO

Two-Color Policer Configuration Overview | 2034

Three-Color Policer Configuration Overview | 2118

Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface | 1937

physical-interface-filter

physical-interface-policer

Example: Configuring a Physical Interface Policer for Aggregate Traffic at a Physical Interface

IN THIS SECTION

- Requirements | 2187
- Overview | 2187
- Configuration | 2188
- Verification | 2194

This example shows how to configure a single-rate two-color policer as a physical interface policer.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

- Topology | 2188

A *physical interface policer* specifies rate-limiting for aggregate traffic, which encompasses all protocol families and logical interfaces configured on a physical interface, even if the interfaces belong to different routing instances.

You can apply a physical interface policer to Layer 3 input or output traffic only by referencing the policer from a stateless firewall filter that is configured for specific a specific protocol family (not for `family any`) and configured as a physical interface filter. You configure the filter terms with match conditions that select the types of packets you want to rate-limit, and you specify the physical interface policer as the action to apply to matched packets.

NOTE: Physical interface policers/filters are not supported for list filters.

Topology

The physical interface policer in this example, `shared-policer-A`, rate-limits to 10,000,000 bps and permits a maximum burst of traffic of 500,000 bytes. You configure the policer to discard packets in nonconforming flows, but you could instead configure the policer to re-mark nonconforming traffic with a forwarding class, a packet loss priority (PLP) level, or both.

To be able to use the policer to rate-limit IPv4 traffic, you reference the policer from an IPv4 physical interface filter. For this example, you configure the filter to pass the policer IPv4 packets that meet either of the following match terms:

- Packets received through TCP and with the IP precedence fields `critical-ecp` (0xa0), `immediate` (0x40), or `priority` (0x20)
- Packets received through TCP and with the IP precedence fields `internet-control` (0xc0) or `routine` (0x00)

You could also reference the policer from physical interface filters for other protocol families.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **2189**
- Configuring the Logical Interfaces on the Physical Interface | **2189**
- Configuring a Physical Interface Policer | **2190**
- Configuring an IPv4 Physical Interface Filter | **2191**
- Applying the IPv4 Physical interface Filter to Reference the Physical Interface Policers | **2193**

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Use the CLI Editor in Configuration Mode" on page 1889.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the [edit] hierarchy level.

```
set interfaces so-1/0/0 unit 0 family inet address 192.168.1.1/24
set interfaces so-1/0/0 unit 0 family vpls
set interfaces so-1/0/0 unit 1 family mpls
set firewall policer shared-policer-A physical-interface-policer
set firewall policer shared-policer-A if-exceeding bandwidth-limit 100m burst-size-limit 500k
set firewall policer shared-policer-A then discard
set firewall family inet filter ipv4-filter physical-interface-filter
set firewall family inet filter ipv4-filter term tcp-police-1 from precedence [ critical-ecp
immediate priority ]
set firewall family inet filter ipv4-filter term tcp-police-1 from protocol tcp
set firewall family inet filter ipv4-filter term tcp-police-1 then policer shared-policer-A
set firewall family inet filter ipv4-filter term tcp-police-2 from precedence [ internet-control
routine ]
set firewall family inet filter ipv4-filter term tcp-police-2 from protocol tcp
set firewall family inet filter ipv4-filter term tcp-police-2 then policer shared-policer-A
set interfaces so-1/0/0 unit 0 family inet filter input ipv4-filter
```

Configuring the Logical Interfaces on the Physical Interface

Step-by-Step Procedure

To configure the logical interfaces on the physical interface:

1. Enable configuration of logical interfaces.

```
[edit]
user@host# edit interfaces so-1/0/0
```

2. Configure protocol families on logical unit 0.

```
[edit interfaces so-1/0/0]
user@host# set unit 0 family inet address 192.168.1.1/24
user@host# set unit 0 family vpls
```

3. Configure protocol families on logical unit 1.

```
[edit interfaces so-1/0/0]
user@host# set unit 1 family mpls
```

Results

Confirm the configuration of the firewall filter by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
        family inet {
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

Configuring a Physical Interface Policer

Step-by-Step Procedure

To configure a physical interface policer:

1. Enable configuration of the two-color policer.

```
[edit]
user@host# edit firewall policer shared-policer-A
```

2. Configure the type of two-color policer.

```
[edit firewall policer shared-policer-A]
user@host# set physical-interface-policer
```

3. Configure the traffic limits and the action for packets in a nonconforming traffic flow.

```
[edit firewall policer shared-policer-A]
user@host# set if-exceeding bandwidth-limit 100m burst-size-limit 500k
user@host# set then discard
```

For a physical interface filter, the actions you can configure for packets in a nonconforming traffic flow are to discard the packets, assign a forwarding class, assign a PLP value, or assign both a forwarding class and a PLP value.

Results

Confirm the configuration of the policer by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
policer shared-policer-A {
    physical-interface-policer;
    if-exceeding {
        bandwidth-limit 100m;
        burst-size-limit 500k;
    }
    then discard;
}
```

Configuring an IPv4 Physical Interface Filter

Step-by-Step Procedure

To configure a physical interface policer as the action for terms in an IPv4 physical interface policer:

1. Configure a standard stateless firewall filter under a specific protocol family.

```
[edit]
user@host# edit firewall family inet filter ipv4-filter
```

   You cannot configure a physical interface firewall filter for `family any`.

2. Configure the filter as a physical interface filter so that you can apply the physical interface policer as an action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set physical-interface-filter
```

3. Configure the first term to match IPv4 packets received through TCP with the IP precedence fields `critical-ecp`, `immediate`, or `priority` and to apply the physical interface policer as a filter action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set term tcp-police-1 from precedence [ critical-ecp immediate priority ]
user@host# set term tcp-police-1 from protocol tcp
user@host# set term tcp-police-1 then policer shared-policer-A
```

4. Configure the first term to match IPv4 packets received through TCP with the IP precedence fields `internet-control` or `routine` and to apply the physical interface policer as a filter action.

```
[edit firewall family inet filter ipv4-filter]
user@host# set term tcp-police-2 from precedence [ internet-control routine ]
user@host# set term tcp-police-2 from protocol tcp
user@host# set term tcp-police-2 then policer shared-policer-A
```

Results

Confirm the configuration of the firewall filter by entering the `show firewall` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
```

```
    filter ipv4-filter {
        physical-interface-filter;
        term tcp-police-1 {
            from {
                precedence [ critical-ecp immediate priority ];
                protocol tcp;
            }
            then policer shared-policer-A;
        }
        term tcp-police-2 {
            from {
                precedence [ internet-control routine ];
                protocol tcp;
            }
            then policer shared-policer-A;
        }
    }
}
policer shared-policer-A {
    physical-interface-policer;
    if-exceeding {
        bandwidth-limit 100m;
        burst-size-limit 500k;
    }
    then discard;
}
```

Applying the IPv4 Physical interface Filter to Reference the Physical Interface Policers

Step-by-Step Procedure

To apply the physical interface filter so it references the physical interface policers:

1. Enable configuration of IPv4 on the logical interface.

```
[edit]
user@host# edit interfaces so-1/0/0 unit 0 family inet
```

2. Apply the IPv4 physical interface filter in the input direction.

```
[edit interfaces so-1/0/0 unit 0 family inet]
user@host# set filter input ipv4-filter
```

Results

Confirm the configuration of the firewall filter by entering the `show interfaces` configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
so-1/0/0 {
    unit 0 {
        family inet {
            filter {
                input ipv4-filter;
            }
            address 192.168.1.1/24;
        }
        family vpls;
    }
    unit 1 {
        family mpls;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Displaying the Firewall Filters Applied to an Interface | 2195
- Displaying the Number of Packets Processed by the Policer at the Logical Interface | 2195

Confirm that the configuration is working properly.

Displaying the Firewall Filters Applied to an Interface

Purpose

Verify that the firewall filter `ipv4-filter` is applied to the IPv4 input traffic at logical interface `so-1/0/0.0`.

Action

Use the `show interfaces statistics` operational mode command for logical interface `so-1/0/0.0`, and include the `detail` option. In the **Protocol inet** section of the command output, the **Input Filters** field shows that the firewall filter `ipv4-filter` is applied in the input direction.

```
user@host> show interfaces statistics so-1/0/0 detail
  Logical interface so-1/0/0.0 (Index 79) (SNMP ifIndex 510) (Generation 149)
    Flags: Hardware-Down Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
    Protocol inet, MTU: 4470, Generation: 173, Route table: 0
      Flags: Sendbcast-pkt-to-re, Protocol-Down
      Input Filters: ipv4-filter
      Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
        Destination: 10.39/16, Local: 10.39.1.1, Broadcast: 10.39.255.255, Generation: 163
```

Displaying the Number of Packets Processed by the Policer at the Logical Interface

Purpose

Verify the traffic flow through the logical interface and that the policer is evaluated when packets are received on the logical interface.

Action

Use the `show firewall` operational mode command for the filter you applied to the logical interface.

```
user@host> show firewall filter ipv4-filter
Filter: ipv4-filter
Policers:
Name                                                Packets
```

```
shared-policer-A-tcp-police-1                    32863
shared-policer-A-tcp-police-2                     3870
```

The command output displays the name of policer (`shared-policer-A`), the name of the filter term (`police-1`) under which the policer action is specified, and the number of packets that matched the filter term. This is only the number of out-of-specification (out-of-spec) packet counts, not all packets policed by the policer.

SEE ALSO

- Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**
- Firewall Filter Match Conditions Based on Bit-Field Values | **976**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**
- Two-Color Policer Configuration Overview | **2034**
- Physical Interface Policer Overview

RELATED DOCUMENTATION

- Firewall Filter Match Conditions Based on Numbers or Text Aliases | **974**
- Firewall Filter Match Conditions Based on Bit-Field Values | **976**
- Firewall Filter Match Conditions Based on Address Fields | **982**
- Firewall Filter Match Conditions Based on Address Classes | **992**
- Two-Color Policer Configuration Overview | **2034**
- Three-Color Policer Configuration Overview | **2118**
- Guidelines for Applying Traffic Policers | **1934**
- *physical-interface-filter*
- *physical-interface-policer*

CHAPTER 34

Configuring Policers on Switches

IN THIS CHAPTER

- Overview of Policers | **2198**
- Traffic Policer Types | **2206**
- Understanding the Use of Policers in Firewall Filters | **2210**
- Understanding Tricolor Marking Architecture | **2215**
- Configuring Policers to Control Traffic Rates (CLI Procedure) | **2215**
- Configuring Tricolor Marking Policers | **2218**
- Understanding Policers with Link Aggregation Groups | **2221**
- Understanding Color-Blind Mode for Single-Rate Tricolor Marking | **2222**
- Understanding Color-Aware Mode for Single-Rate Tricolor Marking | **2222**
- Understanding Color-Blind Mode for Two-Rate Tricolor Marking | **2225**
- Understanding Color-Aware Mode for Two-Rate Tricolor Marking | **2225**
- Example: Using Two-Color Policers and Prefix Lists | **2228**
- Example: Using Policers to Manage Oversubscription | **2232**
- Assigning Forwarding Classes and Loss Priority | **2234**
- Configuring Color-Blind Egress Policers for Medium-Low PLP | **2237**
- Configuring Two-Color and Three-Color Policers to Control Traffic Rates | **2237**
- Verifying That Two-Color Policers Are Operational | **2241**
- Verifying That Three-Color Policers Are Operational | **2242**
- Troubleshooting Policer Configuration | **2243**
- Troubleshooting Policer Configuration | **2247**

Overview of Policers

IN THIS SECTION

- Policer Overview | 2198
- Policer Types | 2201
- Policer Actions | 2202
- Policer Colors | 2203
- Filter-Specific Policers | 2203
- Suggested Naming Convention for Policers | 2203
- Policer Counters | 2204
- Policer Algorithms | 2204
- How Many Policers Are Supported? | 2205
- Policers Can Limit Egress Firewall Filters | 2205

A switch polices traffic by limiting the input or output transmission rate of a class of traffic according to user-defined criteria. Policing (or rate-limiting) traffic allows you to control the maximum rate of traffic sent or received on an interface and to provide multiple priority levels or classes of service.

Policing is also an important component of firewall filters. You can achieve policing by including policers in *firewall filter* configurations.

Policer Overview

You use policers to apply limits to traffic flow and set consequences for packets that exceed these limits —usually applying a higher loss priority—so that if packets encounter downstream congestion, they can be discarded first. Policers apply only to unicast packets.

Policers provide two functions: metering and marking. A policer meters (measures) each packet against traffic rates and burst sizes that you configure. It then passes the packet and the metering result to the marker, which assigns a packet loss priority that corresponds to the metering result. Figure 89 on page 2200 illustrates this process.

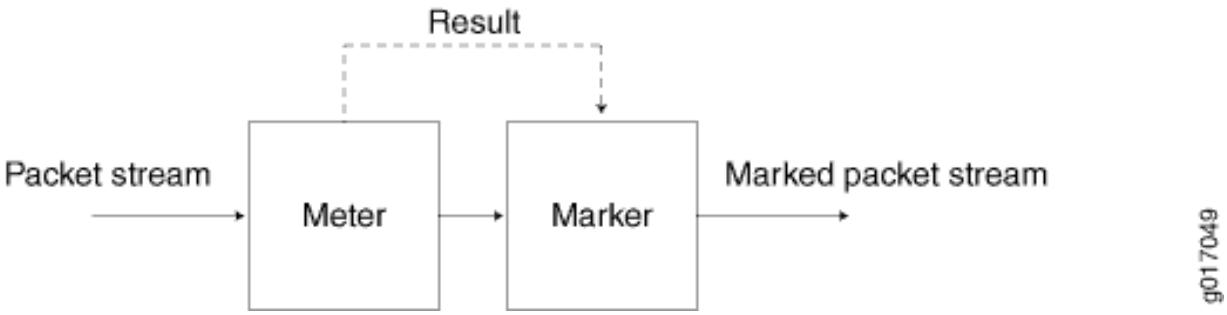
NOTE: A policer restricts traffic at the configured transmission rate per PFE. In QFX10016, QFX10002, QFX10002-60C, and QFX10008 switches, when aggregated ethernet (AE) interface

bundles span multiple PFEs, the overall transmission rate of the policer for the subscriber could exceed the configured transmission rate of the policer (depending on the number of PFEs involved).

As an example:

- Policer with bandwidth-limit 100 mbps configured on an AE interface that has member links xe-1/0/0 (fpc1-pfe0) and xe-1/0/30 (fpc1-pfe1) . Here, the two member links belong to FPC1, but are on different PFEs. When the policer is applied to the AE interface, this will result in a total bandwidth of 200 Mbps as policer is configured for two PFEs.
- Policer with bandwidth-limit 100 mbps configured on an AE interface that has member links xe-1/0/0 (fpc1-pfe0), et-2/0/1 (fpc2-pfe1) and xe-2/0/18:0 (fpc2-pfe2) . Here, one member link belongs to FPC1 and PFE0 on this FPC. Thes rest two member links belong to FPC2, but different PFEs. When the policer is applied to the AE interface, this will result in a total bandwidth of 300 Mbps as policer is configured for three PFEs.
- Policer with bandwidth-limit 100 mbps configured on an AE interface that has member links xe-1/0/0 and xe-1/0/1 on a single PFE (fpc1-pfe0) . Here, the member links belong to FPC1 and to the same PFE. When the policer is applied to the AE interface, this will result in a total bandwidth of 100 Mbps as policer is configured on a per PFE basis.

Figure 89: Flow of Tricolor Marking Policer Operation



After you name and configure a policer, you can use it by specifying it as an action in one or more firewall filters.

Policer Types

A switch supports three types of policers:

- Single-rate two-color marker—A two-color policer (or "policer" when used without qualification) meters the traffic stream and classifies packets into two categories of packet loss priority (PLP) according to a configured bandwidth and burst-size limit. You can mark packets that exceed the bandwidth and burst-size limit with a specified PLP or simply discard them.

  You can specify this type of policer in an ingress or egress firewall.

  > **NOTE**: A two-color policer is most useful for metering traffic at the port (physical interface) level.

- Single-rate three-color marker—This type of policer is defined in RFC 2697, *A Single Rate Three Color Marker*, as part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a Differentiated Services (DiffServ) environment. This type of policer meters traffic based on one rate—the configured committed information rate (CIR) as well as the committed burst size (CBS) and the excess burst size (EBS). The CIR specifies the average rate at which bits are admitted to the switch. The CBS specifies the usual burst size in bytes and the EBS specifies the maximum burst size in bytes. The EBS must be greater than or equal to the CBS, and neither can be 0.

  You can specify this type of policer in an ingress or egress firewall.

  > **NOTE**: A single-rate three-color marker (TCM) is most useful when a service is structured according to packet length and not peak arrival rate.

- Two-rate three-color marker—This type of policer is defined in RFC 2698, *A Two Rate Three Color Marker*, as part of an assured forwarding per-hop-behavior classification system for a Differentiated Services environment. This type of policer meters traffic based on two rates—the CIR and peak information rate (PIR) along with their associated burst sizes, the CBS and peak burst size (PBS). The PIR specifies the maximum rate at which bits are admitted to the network and must be greater than or equal to the CIR.

  You can specify this type of policer in an ingress or egress firewall.

NOTE: A two-rate three-color policer is most useful when a service is structured according to arrival rates and not necessarily packet length.

See Table 120 on page 2202 for information about how metering results are applied for each of these policer types.

Policer Actions

Policer actions are implicit or explicit and vary by policer type. *Implicit* means that Junos OS assigns the loss priority automatically. Table 120 on page 2202 describes the policer actions.

Table 120: Policer Actions

Policer	Marking	Implicit Action	Configurable Action
Single-rate two-color	Green (conforming)	Assign low loss priority	None
	Red (nonconforming)	None	Discard
Single-rate three-color	Green (conforming)	Assign low loss priority	None
	Yellow (above the CIR and CBS)	Assign medium-high loss priority	None
	Red (above the EBS)	Assign high loss priority	Discard
Two-rate three-color	Green (conforming)	Assign low loss priority	None
	Yellow (above the CIR and CBS)	Assign medium-high loss priority	None
	Red (above the PIR and PBS)	Assign high loss priority	Discard

NOTE: If you specify a policer in an egress *firewall filter*, the only supported action is `discard`.

Policer Colors

Single-rate and two-rate three-color policers can operate in two modes:

- Color-blind—In color-blind mode, the three-color policer assumes that all packets examined have not been previously marked or metered. In other words, the three-color policer is “blind” to any previous coloring a packet might have had.
- Color-aware—In color-aware mode, the three-color policer assumes that all packets examined have been previously marked or metered. In other words, the three-color policer is “aware” of the previous coloring a packet might have had. In color-aware mode, the three-color policer can increase the PLP of a packet but cannot decrease it. For example, if a color-aware three-color policer meters a packet with a medium PLP marking, it can raise the PLP level to high but cannot reduce the PLP level to low.

Filter-Specific Policers

You can configure policers to be filter-specific, which means that Junos OS creates only one policer instance regardless of how many times the policer is referenced. When you do this on some QFX switches, rate limiting is applied in aggregate, so if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, the total bandwidth allowed by the filter is 1 Gbps. However, the behavior of a filter-specific policer is affected by how the firewall filter terms that reference the policer are stored in TCAM. If you create a filter-specific policer and reference it in multiple firewall filter terms, the policer allows more traffic than expected if the terms are stored in different TCAM slices. For example, if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms that are stored in three separate memory slices, the total bandwidth allowed by the filter is 3 Gbps, not 1 Gbps. (This behavior does not occur in QFX10000 switches.)

To prevent this unexpected behavior from occurring, use the information about TCAM slices presented in "Planning the Number of Firewall Filters to Create" on page 1721 to organize your configuration file so that all the firewall filter terms that reference a given filter-specific policer are stored in the same TCAM slice.

Suggested Naming Convention for Policers

We recommend that you use the naming convention *policertypeTCM#-color type* when configuring three-color policers and *policer#* when configuring two-color policers. TCM stands for three-color marker. Because policers can be numerous and must be applied correctly to work, a simple naming convention

makes it easier to apply the policers properly. For example, the first single-rate, color-aware three-color policer configured would be named `srTCM1-ca`. The second two-rate, color-blind three-color configured would be named `trTCM2-cb`. The elements of this naming convention are explained below:

- sr (single-rate)
- tr (two-rate)
- TCM (tricolor marking)
- 1 or 2 (number of marker)
- ca (color-aware)
- cb (color-blind)

Policer Counters

On some QFX switches, each policer that you configure includes an implicit counter that counts the number of packets that exceed the rate limits that are specified for the policer. If you use the same policer in multiple terms—either within the same filter or in different filters—the implicit counter counts all the packets that are policed in all of these terms and provides the total amount. (This does not apply to QFX10000 switches.) If you want to obtain separate packet counts for each term on an affected switch, use these options:

- Configure a unique policer for each term.
- Configure only one policer, but use a unique, explicit counter in each term.

Policer Algorithms

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* in allowing a certain amount of bursty traffic before it starts discarding packets.

NOTE: In an environment of light bursty traffic, QFX5200 might not replicate all multicast packets to two or more downstream interfaces. This occurs only at a line rate burst—if traffic is consistent, the issue does not occur. In addition, the issue occurs only when packet size increases beyond 6k in a one gigabit traffic flow.

How Many Policers Are Supported?

QFX10000 switches support 8K policers (all policer types). QFX5100 and QFX5200 switches support 1535 ingress policers and 1024 egress policers (assuming one policer per firewall filter term). QFX5110 switches support 6144 ingress policers and 1024 egress policers (assuming one policer per firewall filter term).

QFX3500 and QFX3600 standalone switches and QFabric Node devices support the following numbers of policers (assuming one policer per firewall filter term):

- Two-color policers used in ingress firewall filters: 767
- Three-color policers used in ingress firewall filters: 767
- Two-color policers used in egress firewall filters: 1022
- Three-color policers used in egress firewall filters: 512

Policers Can Limit Egress Firewall Filters

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
    - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.

- Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

RELATED DOCUMENTATION

Understanding Color-Blind Mode for Single-Rate Tricolor Marking | 2222

Understanding Color-Blind Mode for Two-Rate Tricolor Marking | 2225

Understanding Color-Aware Mode for Single-Rate Tricolor Marking | 2222

Understanding Color-Aware Mode for Two-Rate Tricolor Marking | 2225

Configuring Two-Color and Three-Color Policers to Control Traffic Rates | 2237

Traffic Policer Types

IN THIS SECTION

- Single-Rate Two-Color Policers | 2207
- Three-Color Policers | 2207
- Two-Color and Three-Color Policer Options | 2208

Single-Rate Two-Color Policers

You can use a single-rate two-color policer, or “policer” when used without qualification, to rate-limit a traffic flow to an average bits-per-second arrival rate (specified by the single specified bandwidth limit) while allowing bursts of traffic for short periods (controlled by the single specified burst-size limit). This type of policer categorizes a traffic flow as either green (conforming) or red (nonconforming). Packets in a green flow are implicitly set to a **low** loss priority and then transmitted. Packets in a red flow are handled according to actions specified in the policer configuration. Packets in a red flow can be marked —set to a specified forwarding class, set to a specified loss priority, or both—or they can be discarded.

A single-rate two-color policer is most useful for metering traffic at the port (physical interface) level.

Basic Single-Rate Two-Color Policer

You can apply a basic single-rate two-color policer to Layer 3 traffic in either of two ways: as an interface policer or as a firewall filter policer. You can apply the policer as an *interface policer*, meaning that you apply the policer directly to a logical interface at the protocol family level. If you want to apply the policer to selected packets only, you can apply the policer as a *firewall filter policer*, meaning that you reference the policer in a stateless firewall filter term and then apply the filter to a logical interface at the protocol family level.

Bandwidth Policer

A bandwidth policer is simply a single-rate two-color policer that is defined using a bandwidth limit specified as a percentage value rather than as an absolute number of bits per second. When you apply the policer (as an interface policer or as a firewall filter policer) to a logical interface at the protocol family level, the effective bandwidth limit is calculated based on either the physical interface media rate or the logical interface configured shaping rate.

Logical Bandwidth Policer

A logical bandwidth policer is a bandwidth policer for which the effective bandwidth limit is calculated based on the logical interface configured shaping rate. You can apply the policer as a firewall filter policer only, and the firewall filter must be configured as an interface-specific filter. When you apply an interface-specific filter to multiple logical interfaces on supported routing platforms, any **count** or **policer** actions act on the traffic stream entering or exiting each individual interface, regardless of the sum of traffic on the multiple interfaces.

Three-Color Policers

The Junos OS supports two types of three-color policers: single-rate and two-rate. The main difference between a single-rate and a two-rate policer is that the single-rate policer allows bursts of traffic for

short periods, while the two-rate policer allows more sustained bursts of traffic. Single-rate policing is implemented using a single token-bucket model, so that periods of relatively low traffic must occur between traffic bursts to allow the token bucket to refill. Two-rate policing is implemented using a dual token-bucket model, which allows bursts of traffic for longer periods.

Single-Rate Three-Color Policers

The single-rate three-color type of policer is defined in RFC 2697, *A Single Rate Three Color Marker*. You use this type of policer to rate-limit a traffic flow to a single rate and three traffic categories (green, yellow, and red). A single-rate three-color policer defines a *committed* bandwidth limit and burst-size limit plus an *excess* burst-size limit. Traffic that conforms to the committed traffic limits is categorized as green (conforming). Traffic that conforms to the bandwidth limit while allowing bursts of traffic as controlled by the excess burst-size limit is categorized as yellow. All other traffic is categorized as red.

A single-rate three-color policer is most useful when a service is structured according to packet length, not peak arrival rate.

Two-Rate Three-Color Policers

The two-rate three-color type of policer is defined in RFC 2698, *A Two Rate Three Color Marker*. You use this type of policer to rate-limit a traffic flow to two rates and three traffic categories (green, yellow, and red). A two-rate three-color policer defines a *committed* bandwidth limit and burst-size limit plus a *peak* bandwidth limit and burst-size limit. Traffic that conforms to the committed traffic limits is categorized as green (conforming). Traffic that exceeds the committed traffic limits but remains below the peak traffic limits is categorized as yellow. Traffic that exceeds the peak traffic limits is categorized as red.

A two-rate three-color policer is most useful when a service is structured according to arrival rates and not necessarily packet length.

Two-Color and Three-Color Policer Options

Both two-color and three-color policers can be configured with the following options:

Logical Interface (Aggregate) Policers

A logical interface policer can be a two-color policer, not a three-color policer. When you apply a logical inteface policer to multiple protocol families on the same logical interface, multiple instances of the policer are created, meaning that traffic for each protocol family is policed separately. You apply a logical interface policer directly to a logical interface configuration (and not by referencing the policer in a stateless firewall filter and then applying the filter to the logical interface).

- You can apply the policer at the interface logical unit level to rate-limit all traffic types, regardless of the protocol family.

  When applied in this manner, the logical interface policer will be used by all traffic types (inet, intet6, etc.) and across all layers (layer 2, layer 3) no matter where the policer is attached on the logical interface.

- You can also apply the policer at the logical interface protocol family level, to rate-limit traffic for a specific protocol family.

You can apply a logical interface policer to unicast traffic only. For information about configuring a stateless firewall filter for flooded traffic, see "*Applying Forwarding Table Filters*" in the "Traffic Sampling, Forwarding, and Monitoring" section of the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

Physical Interface Policers

A physical interface policer can be a two-color or three-color policer. When you apply physical interface policer, to different protocol families on the same logical interface, the protocol families share the same policer instance. This means that rate limiting is performed aggregately for the protocol families for which the policer is applied. This feature enables you to use a single policer instance to perform aggregate policing for different protocol families on the same physical interface. If you want a policer instance to be associated with a protocol family, the corresponding physical interface filter needs to be applied to that protocol family. The policer is not automatically applied to all protocol families configured on the physical interface.

In contrast, with logical interface policers there are multiple separate policer instances.

Policers Applied to Layer 2 Traffic

In addition to hierarchical policing, you can also apply single-rate two-color policers and three-color policers (both single-rate and two-rate) to Layer 2 input or output traffic. You must configure the two-color or three-color policer as a logical interface policer and reference the policer in the interface configuration at the logical unit level, and not at the protocol level. You cannot apply a two-color or three-color policer to Layer 2 traffic as a stateless firewall filter action.

Multifield Classification

Like behavior aggregate (BA) classification, which is sometimes referred to as class-of-service (CoS) value traffic classification, multifield classification is a method of classifying incoming traffic by associating each packet with a forwarding class, a packet loss priority level, or both. The CoS scheduling configuration assigns packets to output queues based on forwarding class. The CoS random early detection (RED) process uses the drop probability configuration, output queue fullness percentage, and packet loss priority to drop packets as needed to control congestion at the output stage.

BA classification and multifield classification use different fields of a packet to perform traffic classification. BA classification is based on a *CoS value* in the IP packet header. Multifield classification can be based on *multiple fields* in the IP packet header, including CoS values. Multifield classification is used instead of BA classification when you need to classify packets based on information in the packet other than the CoS values only. Multifield classification is configured using a stateless firewall filter term that matches on any packet header fields and associates matched packets with a forwarding class, a loss priority, or both. The forwarding class or loss priority can be set by a firewall filter action or by a policer referenced as a firewall filter action.

RELATED DOCUMENTATION

- Controlling Network Access Using Traffic Policing Overview | **1916**
- Order of Policer and Firewall Filter Operations | **1926**
- Two-Color Policer Configuration Overview | **2034**
- Three-Color Policer Configuration Overview | **2118**
- Two-Color Policing at Layer 2 Overview
- Three-Color Policing at Layer 2 Overview

Understanding the Use of Policers in Firewall Filters

IN THIS SECTION

- Policers Overview | **2211**
- Policer Types | **2211**
- Policer Actions | **2212**
- Policer Levels | **2213**
- Color Modes | **2213**
- Naming Conventions for Policers | **2214**

Policing, or rate limiting, is an important component of firewall filters that lets you control the amount of traffic that enters an interface on Juniper Networks EX Series Ethernet Switches. You can achieve policing by including policers in *firewall filter* configurations.

Policers Overview

You can use policers to specify rate limits on traffic. A firewall filter configured with a policer permits only traffic within a specified set of rate limits, thereby providing protection from denial-of-service (DoS) attacks. Traffic that exceeds the rate limits specified by the policer is either discarded immediately or is marked as lower priority than traffic that is within the rate limits. The switch discards the lower-priority traffic when there is traffic congestion.

A policer applies two types of rate limits on 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 an algorithm to enforce a limit on average bandwidth while allowing bursts up to a specified maximum value. You can define specific classes of traffic on an interface and apply a set of rate limits to each class. After you name and configure a policer, it is stored as a template. You can then use the policer in a firewall filter configuration.

On all EX Series switches except Juniper Networks EX8200 Ethernet Switches, each policer that you configure includes an implicit counter that counts the number of packets that exceed the rate limit specified for the policer. Each EX8200 switch contains three global management counters. You must assign ingress policers to these global management counters to obtain policer statistics. You can assign any number of ingress policers to each global management counter. The policer statistics for each global management counter are the aggregate of the policer statistics for all policers associated with that global management counter.

To get filter-specific packet counts, you must configure a different policer for each firewall filter. Policers give term-specific counts by default.

Policer Types

Switches support three types of policers:

- Single-rate two-color—A two-color policer (sometimes called simply “policer”) meters the traffic stream and classifies packets into two categories of packet loss priority (PLP) according to a configured bandwidth and burst-size limit. You can mark packets that exceed the bandwidth and burst-size limit or simply discard them. A two-color policer is most useful for metering traffic at the port (physical interface) level.
- Single-rate three-color—This type of policer is defined in RFC 2697, A *Single Rate Three Color Marker*, as part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a Differentiated Services (DiffServ) environment. This type of policer meters traffic based on the configured committed information rate (CIR), committed burst size (CBS), and the excess burst size

(EBS). Traffic is marked as belonging to one of three categories (green, yellow, or red) based on whether the packets are arriving at rates that are below the CBS (green), exceed the CBS but not the EBS (yellow), or exceed the EBS (red). A single-rate three-color policer is most useful when a service is structured according to packet size and not according to peak arrival rate.

- Two-rate three-color—This type of policer is defined in RFC 2698, A *Two Rate Three Color Marker*, as part of an assured forwarding (AF) per-hop-behavior (PHB) classification system for a Differentiated Services (DiffServ) environment. This type of policer meters traffic based on the configured CIR and the peak information rate (PIR), along with their associated burst sizes; the CBS, and the peak burst size (PBS). Traffic is marked as belonging to one of three categories (green, yellow, or red) based on packets are arriving at rates that are below the CIR (green), exceed the CIR but not the PIR (yellow), or exceed the PIR (red). A two-rate three-color policer is most useful when a service is structured according to arrival rates and not to packet size.

Policer Actions

Policer actions can be implicit or explicit and vary by policer type. The term implicit means that Junos OS assigns a loss-priority value automatically; explicit means that you configure the action. Table 121 on page 2212 lists policer actions.

Table 121: Policer Actions

Policer	Marking	Implicit Action	Configurable Action
Single-rate two-color	Green (Conforming)	Assign low loss priority	None
	Red (Nonconforming)	None	Assign low or high loss priority, assign a forwarding class, or discard
	Yellow	Not supported	Not supported
Single-rate three-color	Green (Conforming)	Assign low loss priority	None
	Red (Above the EBS)	Assign high loss priority	Discard

Table 121: Policer Actions *(Continued)*

Policer	Marking	Implicit Action	Configurable Action
	Yellow (Exceeds the CBS but not the EBS)	Assign high loss priority **NOTE**: Not supported on EX8200 switches	None **NOTE**: Not supported on EX8200 switches
Two-rate three-color	Green (Conforming)	Assign low loss priority	None
	Red (Above the PIR)	Assign high loss priority	Discard
	Yellow (Exceeds the CIR but not the PIR)	Assign high loss priority **NOTE**: Not supported on EX8200 switches	None **NOTE**: Not supported on EX8200 switches

NOTE: You cannot apply a policer with an action of `forwarding-class` to an output firewall filter.

NOTE: Beginning with Junos OS Release 17.1, on EX4300 switches, you can configure the policer action `loss-priority` to be `low`, `medium-low`, `medium-high`, or `high`.

Policer Levels

You can configure policers at the queue level, *logical interface* level, or Layer 2 (MAC) level. Only a single policer is applied to a packet at the egress queue. The search for policers occurs in this order:

- Queue level
- Logical interface level
- Layer 2 (MAC) level

Color Modes

Tricolor marking (TCM) policers are not bound by a green-yellow-red coloring convention. Packets are marked with low or high PLP bit configurations based on color. Therefore, both three-color policer types

(single-rate and two-rate) extend the functionality of class-of-service (CoS) traffic policing by providing three levels of drop precedence (loss priority) instead of the two normally available in policers. Both single-rate and two-rate three-color policer types can operate in two modes:

- Color-blind—In color-blind mode, the three-color policer operates without reference to whether the examined packets have been previously marked or metered. In other words, the three-color policer is *blind* to any previous coloring a packet might have had.
- Color-aware—In color-aware mode, the three-color policer operates with reference to any previous marking or metering of the examined packets. In other words, the three-color policer is *aware* of the previous coloring a packet might have had. In color-aware mode, the three-color policer can increase the PLP of a packet but can never decrease it. For example, if a color-aware three-color policer meters a packet with a low PLP marking, it can raise the PLP level to high. But it cannot reduce a high PLP level to low.

Naming Conventions for Policers

We recommend you use the naming convention *rate-TCMnumber-colortype* when configuring three-color policers. TCM stands for tricolor marking. Because policers can be numerous and must be applied correctly to work, observing a simple naming convention makes it easier to apply the policers properly.

For example, if you configure a single-rate, three-color, color-aware policer, name it srTCM1-ca. If you configure a two-rate, three-color, color-blind policer, name it trTCM2-cb.

Release History Table

Release	Description
17.1	Beginning with Junos OS Release 17.1, on EX4300 switches, you can configure the policer action `loss-priority` to be `low`, `medium-low`, `medium-high`, or `high`.

RELATED DOCUMENTATION

Firewall Filters for EX Series Switches Overview | **1535**

Understanding Tricolor Marking Architecture | **2215**

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | **1650**

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | **1554**

Understanding Tricolor Marking Architecture

Tricolor marking (TCM) policers provide two functions: metering and marking. A policer meters each packet and passes the packet and the metering result to the marker.

The meter operates in two modes. In the color-blind mode, the meter treats the packet stream as uncolored. Any preset loss priorities are ignored. In the color-aware mode, the meter inspects the packet loss priority (PLP) field, which has been set by an upstream device as high or low; in other words, the PLP field has already been set by a behavior aggregate (BA) or multifield (MF) classifier. The marker changes the PLP of each incoming IP packet according to the results of the meter.

Single-rate TCM is so called because traffic is policed according to one rate—the committed burst rate (CBR)—and two burst sizes: the committed burst size (CBS) and the excess burst size (EBS). The configured information rate (CIR) specifies the average rate at which bits are admitted to the network. The CBS specifies the usual burst size in bytes and the EBS specifies the maximum burst size in bytes for packets that are admitted to the network. The EBS is greater than or equal to the CBS, and neither can be 0. As each packet enters the network, its bytes are counted. Packets that do not exceed the CBS are marked low PLP. Packets that exceed the peak information rate (PIR) are marked high PLP.

Two-rate TCM is so called because traffic is policed according to two rates: the CIR and the PIR. The PIR is greater than or equal to the CIR. The CIR specifies the average rate at which bits are admitted to the network, and the PIR specifies the maximum rate at which bits are admitted to the network. As each packet enters the network, its bits are counted. Bits in packets that do not exceed the CIR have their packets marked low PLP. Bits in packets that exceed the PIR have their packets marked high PLP.

RELATED DOCUMENTATION

Understanding the Use of Policers in Firewall Filters | 2210

Configuring Tricolor Marking Policers | 2218

Configuring Policers to Control Traffic Rates (CLI Procedure)

IN THIS SECTION

- Configuring Policers | 2216
- Specifying Policers in a Firewall Filter Configuration | 2218
- Applying a Firewall Filter That Is Configured with a Policer | 2218

You can configure policers to rate limit traffic on EX Series switches. After you configure a policer, you can include it in an ingress firewall filter configuration.

When you configure a firewall filter, you can specify a policer action for any term or terms within the filter. All traffic that matches a term that contains a policer action goes through the policer that the term references. Each policer that you configure includes an implicit counter. To get term-specific packet counts, you must configure a separate policer for each filter term that requires policing.

NOTE: On all EX Series switches except EX8200 switches, each policer that you configure includes an implicit counter. To ensure term-specific packet counts, configure a policer for each term in the filter that requires policing. For EX8200 switches, configure a policer and associate it with a global management counter using the *counter* option.

The following policer limits apply on a switch:

- A maximum of 512 policers can be configured for port firewall filters.
- A maximum of 512 policers can be configured for VLAN and Layer 3 firewall filters.

If the number of policers in the firewall filter configuration exceeds these limits, the switch returns the following message when you commit the configuration:

```
Cannot assign policers: Max policer limit reached
```

This topic includes these tasks:

Configuring Policers

To configure a policer:

1. Specify the name of the policer:

   ```
   [edit firewall]
    user@switch# set policer policer-one
   ```

   The policer name can include letters, numbers, and hyphens (-) and can contain up to 64 characters.
2. Specify the filter-specific statement to configure a policer to act as a filter-specific policer; else proceed to step 3:

   ```
   [edit firewall]
    user@switch# set policer policer-one filter-specific
   ```

If you do not specify the `filter-specific` statement, the policer acts as a term-specific policer by default.

3. Configure rate limiting for the policer:

   a. Specify the bandwidth limit in bits per second (bps) to control the traffic rate on an interface:

   ```
   [edit firewall policer policer-one]
    user@switch# set if-exceeding bandwidth-limit 300k
   ```

   The range for the bandwidth limit is 1k through 102.3g bps.

   b. Specify the burst-size limit (the maximum allowed burst size in bytes) to control the amount of traffic bursting:

   ```
   [edit firewall policer policer-one]
    user@switch# set if-exceeding burst-size-limit 500k
   ```

   To determine the value for the burst-size limit, multiply the bandwidth of the interface on which the filter is applied by the amount of time to allow a burst of traffic at that bandwidth to occur:

   burst size = (bandwidth) * (allowable time for burst traffic)

   The range for the burst-size limit is 1 through 2,147,450,880 bytes.

4. Specify the policer action **discard** to discard packets that exceed the rate limits:

   ```
   [edit firewall policer]
    user@switch# set policer-one then (Policer Action) discard
   ```

   Discard is the only supported policer action.

5. On EX8200 switches, you must assign a global management counter to the policer to obtain policer statistics:

   ```
   [edit firewall policer]
   user@switch# set policer-one counter counter-id 0
   ```

   In this sample statement, the global management counter ID is **0**. You can assign any number of policers to the global management counter. The policer statistics displayed for each counter are the collective statistics of all policers assigned to that counter.

Specifying Policers in a Firewall Filter Configuration

To reference a policer for a single firewall, configure a filter term that includes the policer action:

```
[edit firewall family ethernet-switching]
user@switch# set filter limit-hosts term term-one from source-address 192.0.2.0/28
users@witch# set filter limit-hosts term term-one then policer policer-one
```

Applying a Firewall Filter That Is Configured with a Policer

A firewall filter that is configured with one or more policer actions, like any other firewall filter, must be applied to a port, VLAN, or Layer 3 interface. For information about applying firewall filters, see the sections on applying firewall filters in "Configuring Firewall Filters (CLI Procedure)" on page 1638.

RELATED DOCUMENTATION

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Configuring Firewall Filters (CLI Procedure) | 1638

Verifying That Policers Are Operational | 1699

Understanding the Use of Policers in Firewall Filters | 2210

Configuring Tricolor Marking Policers

IN THIS SECTION

- Configuring a Tricolor Marking Policer | 2219
- Applying Tricolor Marking Policers to Firewall Filters | 2220

You can rate-limit traffic on EX Series switches by configuring a policer and specifying it as an action modifier for a term in a firewall filter. By default, if you specify the same policer in multiple terms, Junos OS creates a separate policer instance for each term and applies rate limiting separately for each instance. For example, if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, each policer instance enforces a 1-Gbps limit. In this case, the total bandwidth allowed by the filter is 3 Gbps.

You can also configure a policer to be filter-specific, which means that Junos OS creates only one policer instance regardless of how many times the policer is referenced. When you do this, rate limiting is applied in aggregate, so if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, the total bandwidth allowed by the filter is 1 Gbps.

This topic describes how to configure single-rate and two-rate tricolor marking (TCM) policers, also known as single-rate and two-rate three-color policers. If you want to configure a single-rate two-color policer (also known just as a "policer"), see "Configuring Policers to Control Traffic Rates (CLI Procedure)" on page 2215.

Configuring a Tricolor Marking Policer

A tricolor marking policer polices traffic on the basis of metering rates, including the configured information rate (CIR), the peak information rate (PIR), their associated burst sizes, and any policing actions configured for the traffic. With tri-color marking, you can configure traffic policing according to two separate modes—color-blind and color-aware. In color-blind mode, the current packet loss priority (PLP) value is ignored. In color-aware mode, the current PLP values are considered by the policer, and the policer can increase those values but cannot decrease them.

To configure a tricolor marking (TCM) policer:

1. Specify the name of the policer and (optionally) whether to automatically discard packets with high loss priority (PLP):

```
[edit firewall]
 user@switch# set three-color-policer policer-name
 user@switch# set three-color-policer policer-name action loss-priority high then discard
```

2. Specify the policer as either single-rate or two-rate and as color-aware or color-blind:

```
[edit firewall three-color-policer policer-name]
 user@switch# set rate mode
```

For example:

```
[edit firewall three-color-policer srTCm-1a]
 user@switch# set single-rate color-aware
[edit firewall three-color-policer trTCM2-cb]
 user@switch# set two-rate color-blind
```

3. For a single-rate TCM policer, configure the CIR, committed burst size (CBS), and excess burst size (EBS):

```
[edit firewall three-color-policer policer-name single-rate]
 user@switch# set committed-information-rate bps
 user@switch# set committed-burst-size bytes
 user@switch# set excess-burst-size bytes
```

4. For a two-rate TCM policer, configure the CIR, CBS, PIR, and peak burst size (PBS):

```
[edit firewall three-color-policer policer-name single-rate]
 user@switch# set committed-information-rate bps
 user@switch# set committed-burst-size bytes
 user@switch# set peak-information-rate bps
 user@switch# set peak-burst-size bytes
```

Applying Tricolor Marking Policers to Firewall Filters

To rate-limit traffic by applying a tricolor marking (TCM) policer to a firewall filter:

```
[edit firewall family family filter filter-name term term-name then]
user@switch# set three-color-policer rate stTCM1-ca
```

For example:

```
[edit firewall family inet filter test1 term term1 then]
user@switch# set three-color-policer single-rate policer1
```

You must include either the `single-rate` statement or the `two-rate` statement in the reference to the policer in the firewall filter configuration, and this statement must match the configured TCM policer. Otherwise, an error message appears in the configuration listing.

For example, if you configure **srTCM1-ca** as a single-rate TCM policer and try to apply it as a two-rate policer, the following message appears:

```
[edit firewall]
user@switch# show three-color-policer srTCM1-ca
single-rate {
    color-aware;
    . . .
```

```
}
user@switch# show filter TESTER
term A {
    then {
        three-color-policer {
            ##
            ## Warning: Referenced two-rate policer does not exist
            ##
            two-rate srTCM;
        }
    }
}
```

RELATED DOCUMENTATION

- Understanding Tricolor Marking Architecture | 2215
- Understanding the Use of Policers in Firewall Filters | 2210

Understanding Policers with Link Aggregation Groups

If you apply a policer to a link aggregation group (LAG) on a standalone switch or QFabric node, the policer applies to all the interfaces in the LAG in aggregate. For example, if you configure a policer to rate-limit at 1 Gbps and apply the policer (by using a *firewall filter*) to a LAG that has two member interfaces on a single switch or node, the total allowed throughput for both members is 1 Gbps.

If you apply a policer to a LAG that has members on different nodes in a QFabric network Node group or redundant server Node group, the configured rate applies to the interface on each node. For example, if you configure a policer to rate-limit at 1 Gbps and apply the policer to a LAG that has one member on server node A and one member on server node B, the allowed throughput for each member is 1 Gbps, for a total allowed throughput of 2 Gbps.

RELATED DOCUMENTATION

- Overview of Policers | 2198
- Configuring Two-Color and Three-Color Policers to Control Traffic Rates | 2237

Understanding Color-Blind Mode for Single-Rate Tricolor Marking

With the color-blind mode of single-rate tricolor marking, all packets are evaluated against the CBS. If a packet exceeds the CBS, it is evaluated against the EBS. In color-blind mode, the policer supports three loss priorities only: low, medium-high, and high.

Packets that exceed the CBS but are below the EBS are marked yellow (medium-high). Packets that exceed the EBS are marked red (high), as shown in Table 122 on page 2222.

Table 122: Color-Blind Mode TCM Color-to-PLP Mapping

Color	PLP	Meaning
Green	**low**	Conforming.
Yellow	**medium-high**	Packet exceeds the CIR and CBS but does not exceed the EBS.
Red	**high**	Packet exceeds the EBS.

RELATED DOCUMENTATION

Overview of Policers | **2198**

Configuring Color-Blind Egress Policers for Medium-Low PLP | **2237**

Understanding Color-Aware Mode for Single-Rate Tricolor Marking

IN THIS SECTION

- Summary of PLP Changes | **2223**

In color-aware mode, the treatment the packet receives depends on its classification. Marking can increase a preassigned PLP but cannot decrease it.

Summary of PLP Changes

Table 123 on page 2223 shows how a packet's incoming priority can be modified with single-rate marking.

Table 123: Color-Aware Mode Single-Rate PLP Mapping

Incoming PLP	Packet Metered Against	Possible Cases	Outgoing PLP
low	CIR, CBS, and EBS	Conforming	**low**
		Packet exceeds the CIR and CBS but does not exceed the EBS.	**medium-high**
		Packet exceeds the EBS.	**high**
medium-low	EBS only	Packet does not exceed the EBS.	**medium-low**
		Packet exceeds the EBS.	**high**
medium-high	EBS only	Packet does not exceed the EBS.	**medium-high**
		Packet exceeds the EBS.	**high**
high	Not metered by the policer.	All cases.	**high**

The following sections describe single-rate color-aware PLP mapping in more detail.

Effect on Green Packets (Low PLP)

Packets belonging to the green class have already been marked by a classifier with low PLP. The marking policer can leave the PLP unchanged or increase it to medium-high or high, so these packets are therefore metered against both the CBS and the EBS. For example, if a behavior aggregate or multifield classifier marks a packet with low PLP and the two-rate TCM policer is in color-aware mode, the output loss priority is as follows:

- If the rate of traffic flow is less than the CIR, packets remain marked as low PLP.
- If bursts exceed the CBS but not the EBS, some of the packets are marked as medium-high PLP, and some of the packets remain marked as low PLP.

- If bursts exceed the EBS, some of the packets are marked as high PLP, and some of the packets remain marked as low PLP.

Effect on Yellow Packets (Medium PLP)

Packets belonging to the yellow class have already been marked by a classifier with medium-low or medium-high PLP. The marking policer can leave the PLP unchanged or increase it to high, so these packets are therefore metered against the EBS only. For example, if a behavior aggregate or multifield classifier marks a packet with medium-low PLP and the two-rate TCM policer is in color-aware mode, the output loss priority is as follows:

- If the rate of traffic flow is less than the CBS, the packets remain marked as medium-low PLP.
- If the rate of traffic flow is greater than the CBS but less than the EBS, the packets remain marked as medium-low PLP.
- If the rate of traffic flow is greater than the EBS, some of the packets are marked as high PLP and some remain marked as medium-low PLP.

If a BA or multifield classifier marks a packet with medium-high PLP and the two-rate TCM policer is in color-aware mode, the policer assigns output loss priority as follows:

- If the rate of traffic flow is less than the CBS, the packets remain marked as medium-high PLP.
- If the rate of traffic flow is greater than the CBS but less than the EBS, the packets remain marked as medium-high PLP.
- If the rate of traffic flow is greater than the EBS, some of the packets are marked as high PLP and some remain marked as medium-high PLP.

Effect on Red Packets (High PLP)

Packets belonging to the red class have already been marked by a classifier with high PLP. Because the policer cannot decrease the PLP, it does not change it, and these packets are not metered against the CBS or the EBS.

RELATED DOCUMENTATION

Overview of Policers | **2198**

Configuring Color-Blind Egress Policers for Medium-Low PLP | **2237**

Understanding Color-Blind Mode for Two-Rate Tricolor Marking

With the color-blind mode of two-rate tricolor marking, all packets are evaluated against the committed information rate (CIR). If a packet exceeds the CIR, it is evaluated against the peak information rate (PIR). Packets that exceed the CIR but are below the PIR are marked yellow (medium-high). Packets that exceed the PIR are marked red (high).

Table 124: Color-Blind Mode TCM Color-to-PLP Mapping

Color	PLP	Meaning
Green	**low**	Packet does not exceed the CIR.
Yellow	**medium-high**	Packet exceeds the CIR but does not exceed the PIR.
Red	**high**	Packet exceeds the PIR.

RELATED DOCUMENTATION

Overview of Policers | **2198**

Configuring Color-Blind Egress Policers for Medium-Low PLP | **2237**

Understanding Color-Aware Mode for Two-Rate Tricolor Marking

IN THIS SECTION

- Summary of PLP Changes | **2226**
- Effect on Green Packets (Low PLP) | **2226**
- Effect on Yellow Packets (Medium PLP) | **2227**
- Effect on Red Packets (High PLP) | **2227**

In color-aware mode, the treatment the packet receives depends on its classification. Marking can increase the preassigned PLP but cannot decrease it

Summary of PLP Changes

Table 125 on page 2226 shows how a packet's incoming priority can be modified with two-rate marking.

Table 125: Color-Aware Mode Two-Rate PLP Mapping

Incoming PLP	Packet Metered Against	Possible Cases	Outgoing PLP
low	CIR and PIR	Packet does not exceed the CIR.	**low**
		Packet exceeds the CIR but not the PIR.	**medium-high**
		Packet exceeds the PIR.	**high**
medium-low	PIR only	Packet does not exceed the PIR.	**medium-low**
		Packet exceeds the PIR.	**high**
medium-high	PIR only	Packet does not exceed the PIR.	**medium-high**
		Packet exceeds the PIR.	**high**
high	Not metered by the policer.	All cases.	**high**

The following sections describe color-aware two-rate PLP mapping in more detail.

Effect on Green Packets (Low PLP)

Packets belonging to the green class have already been marked by a classifier with low PLP. The marking policer can leave the packet's PLP unchanged or increase the PLP to medium-high or high. These packets are therefore metered against both the CIR and the PIR. For example, if a behavior aggregate or multifield classifier marks a packet with low PLP and the two-rate TCM policer is in color-aware mode, the output loss priority is as follows:

- If the rate of traffic flow is less than the CIR, the packets remain marked as low PLP.

- If the rate of traffic flow is greater than the CIR but less than the PIR, some of the packets are marked as medium-high PLP and some of the packets remain marked as low PLP.
- If the rate of traffic flow is greater than the PIR, some of the packets are marked as high PLP and some of the packets remain marked as low PLP.

Effect on Yellow Packets (Medium PLP)

Packets belonging to the yellow class have already been marked by a classifier with medium-low or medium-high PLP. The marking policer can leave the PLP unchanged or increase it to high. These packets are therefore metered against the PIR only. For example, if a behavior aggregate (BA) or multifield classifier marks a packet with medium-low PLP and the two-rate TCM policer is in color-aware mode, the policer assigns output loss priority as follows:

- If the rate of traffic flow is less than the CIR, the packets remain marked as medium-low PLP.
- If the rate of traffic flow is greater than the CIR but less than the PIR, the packets remain marked as medium-low PLP.
- If the rate of traffic flow is greater than the PIR, some of the packets are marked as high PLP and some of the packets remain marked as medium-low PLP.

If a BA or multifield classifier marks a packet with medium-high PLP and the two-rate TCM policer is in color-aware mode, the policer assigns output loss priority as follows:

- If the rate of traffic flow is less than the CIR, the packets remain marked as medium-high PLP.
- If the rate of traffic flow is greater than the CIR but less than the PIR, the packets remain marked as medium-high PLP.
- If the rate of traffic flow is greater than the PIR, some of the packets are marked as high PLP and some of the packets remain marked as medium-high PLP.

Effect on Red Packets (High PLP)

Packets belonging to the red class have already been marked by a classifier with high PLP. Because the policer cannot decrease the PLP, it does not change it, and these packets are not metered against the CIR or the PIR.

RELATED DOCUMENTATION

Overview of Policers | 2198

Configuring Color-Blind Egress Policers for Medium-Low PLP | 2237

Example: Using Two-Color Policers and Prefix Lists

If you provide specific amounts of bandwidth to internal or external customers, you can use policing to make sure that customers do not consume more bandwidth than they should receive. For example, you might connect many customers to one 10-Gbps interface and want to ensure that none of them congest the interface by using more bandwidth than they have been allotted.

You could accomplish this by creating a two-color policer similar to the following for each customer:

```
firewall {
    policer Limit-Customer-1 {
        if-exceeding {
            bandwidth-limit 100m;
            burst-size-limit 150m;
        }
        then discard;
    }
```

Creating a policer for each customer is clearly not a scalable solution, however. As an alternative, you can create prefix lists that group classes of customers and then create policers for each prefix list. For example, you could create prefix lists such as **Class-A-Customer-Prefixes**, **Class-B-Customer-Prefixes**, and **Class-C-Customer-Prefixes** (at the `[edit policy-options]` hierarchy level) and create the following corresponding policers:

```
firewall {
    policer Class-A {
        if-exceeding {
            bandwidth-limit 100m;
            burst-size-limit 150m;
        }
        then discard;
    }
    policer Class-B {
        if-exceeding {
            bandwidth-limit 75m;
            burst-size-limit 100m;
        }
        then discard;
    }
    policer Class-C {
        if-exceeding {
```

```
            bandwidth-limit 50m;
            burst-size-limit 75m;
        }
        then discard;
    }
}
```

You must create filter terms that specify the prefix lists in their `from` statements and the corresponding policers in their `then` statements similar to the following:

```
firewall
    family inet {
        filter Class-A-Customers {
            term term-1 {
                from {
                    destination-prefix-list {
                        Class-A-Customer-Prefixes;
                    }
                }
                then policer Class-A;
            }
        }
        filter Class-B-Customers {
            term term-1 {
                from {
                    destination-prefix-list {
                        Class-B-Customer-Prefixes;
                    }
                }
                then policer Class-B;
            }
        }
        filter Class-C-Customers {
            term term-1 {
                from {
                    destination-prefix-list {
                        Class-C-Customer-Prefixes;
                    }
                }
                then policer Class-C;
            }
```

```
        }
    }
```

Here are the steps to create this firewall configuration:

1. Create the first policer:

```
[edit firewall]
user@switch# set policer Class-A if-exceeding bandwidth-limit 100m burst-size-limit 150m
user@switch# set policer Class-A then discard
```

2. Create the second policer:

```
[edit firewall]
user@switch# set policer Class-B if-exceeding bandwidth-limit 75m burst-size-limit 100m
user@switch# set policer Class-B then discard
```

3. Create the third policer:

```
[edit firewall]
user@switch# set policer Class-C if-exceeding bandwidth-limit 50m burst-size-limit 75m
user@switch# set policer Class-C then discard
```

4. Create a filter for class A customers:

```
[edit firewall]
user@switch# edit family inet filter Class-A-Customers
```

5. Configure the filter to send packets matching the **Class-A-Customer-Prefixes** prefix list to the **Class-A** policer:

```
[edit firewall family inet filter Class-A-Customers]
user@switch# set term term-1 from source-prefix-list Class-A-Customers
user@switch# set term term-1 then policer Class-A
```

6. Create a filter for class B customers:

```
[edit firewall]
user@switch# edit family inet filter Class-B-Customers
```

7. Configure the filter to send packets matching the **Class-B-Customer-Prefixes** prefix list to the **Class-B** policer:

```
[edit firewall family inet filter Class-B-Customers]
user@switch# set term term-1 from source-prefix-list Class-B-Customers
user@switch# set term term-1 then policer Class-B
```

8. Create a filter for class C customers:

```
[edit firewall]
user@switch# edit family inet filter Class-C-Customers
```

9. Configure the filter to send packets matching the **Class-C-Customer-Prefixes** prefix list to the **Class-C** policer:

```
[edit firewall family inet filter Class-C-Customers]
user@switch# set term term-1 from source-prefix-list Class-C-Customers
user@switch# set term term-1 then policer Class-C
```

10. Apply the filters you created to the appropriate interfaces in the output direction.

NOTE: Note that the implicit deny statement in this filter will block traffic from any source that does not match one of the prefix lists. If you want the filter to allow this traffic, you must include an explicit term for this purpose.

RELATED DOCUMENTATION

Overview of Policers | 2198

prefix-list

Example: Using Policers to Manage Oversubscription

You might want to use a policer when an interface is oversubscribed and you want to control what will happen if congestion occurs. For example, you might have servers connected to a switch as listed in Table 126 on page 2232.

Table 126: Servers Connected to Switch

Server Type	Connection	IP Address
Network application server	1-gigabit interface	10.0.0.1
Authentication server	1-gigabit interface	10.0.0.2
Database server	10-gigabit interface	10.0.0.3

In this example, users access services provided by the network application server, which requests information from the database server as appropriate. When it receives a request from a user, the network application server first contacts the authentication server to verify the user's credentials. When a user is authenticated and the network application server provides the requested service, all the packets sent from the database server to the application server must transit the 1-Gigabit Ethernet interface connected to the application server twice—once on ingress to the application server and again on egress to the user.

The sequence of events for a user session is as follows:

1. A user connects to the application server and requests a service.
2. The application server requests the user's credentials and relays them to the authentication server.
3. If the authentication server verifies the credentials, the application server initiates the requested service.
4. The application server requests the files necessary to meet the user's request from the database server.
5. The database server sends the requested files to the application server.
6. The application server includes the requested files in its response to the user.

Traffic from the database server to the application server might congest the 1-gigabit interface to which that the application server is connected. This congestion might prevent the server from responding to

requests from users and creating new sessions for them. You can use policing to make sure that this does not occur.

To create this firewall configuration, perform the following steps on the database server:

1. Create a policer to drop traffic from the database server to the application server if it exceeds certain limits:

```
[edit firewall]
user@switch# set policer Database-Egress-Policer if-exceeding bandwidth-limit 400 burst-size-limit 500m
user@switch# set policer Database-Egress-Policer then discard
```

2. Create a filter to examine traffic from the database server to the application server:

```
[edit firewall]
user@switch# edit family inet filter Database-Egress-Filter
```

3. Configure the filter to apply the policer to traffic egressing the database server and destined for the application server:

```
[edit firewall family inet filter Database-Egress-Filter]
user@switch# set term term-1 from destination-address 10.0.0.1
user@switch# set term term-1 then policer Database-Egress-Policer
```

4. If required, configure a term to allow traffic from the database server to other destinations (otherwise the traffic will be dropped by the implicit deny statement):

```
[edit firewall family inet filter Database-Egress-Filter]
user@switch# set term term-2 then accept
```

   Note that omitting a `from` statement causes the term to match all packets, which is the desired behavior.

5. Install the egress filter as an output filter on the database server interface that is connected the application server:

```
[edit interfaces]
user@switch# set xe-0/0/3 unit 0 family inet filter output Database-Egress-Filter
```

Here is how the final configuration would appear:

```
firewall {
    policer Database-Egress-Policer {
        if-exceeding {
            bandwidth-limit 400;
            burst-size-limit 500m;
        }
        then discard;
    }
    family inet {
        filter Database-Egress-Filter {
            term term-1 {
                from {
                    destination-address {
                        10.0.0.1/24;
                    }
                }
                then policer Database-Egress-Policer;
            }
            term term-2 {  # If required, include this term so that traffic from the database
server to other destinations is allowed.
                then accept;
            }
        }
    }
]
```

RELATED DOCUMENTATION

Overview of Policers | 2198

Assigning Forwarding Classes and Loss Priority

You can configure firewall filters to assign packet loss priority (PLP) and forwarding classes so that if congestion occurs, the marked packets can be dropped according to the priority you set. The valid match conditions are one or more of the six packet header fields: destination address, source address, IP protocol, source port, destination port, and DSCP. In other words, you can set the forwarding class and

the PLP for each packet entering or an interface with a specific destination address, source address, IP protocol, source port, destination port, or DSCP.

NOTE: Junos OS assigns forwarding classes and PLP on ingress only. Do not use a filter that assigns forwarding classes or PLP as an egress filter.

When tricolor marking is enabled, a switch supports four PLP designations: **low**, **medium-low**, **medium-high**, and **high**. You can also specify any of the forwarding classes listed in Table 127 on page 2235

Table 127: Unicast Forwarding Classes

Unicast Forwarding Class	For CoS Traffic Type
be	Best-effort traffic
no-loss	Guaranteed delivery for TCP traffic
fcoe	Guaranteed delivery for Fibre Channel over Ethernet (FCoE) traffic
nc	Network-control traffic

To assign forwarding classes in firewall filters:

1. Configure the family address type and filter name:

```
[edit]
user@switch# edit firewall family ethernet-switching filter ingress-filter
```

2. Configure the terms of the filter as appropriate, including the **forwarding-class** and **loss-priority** action modifiers. For example, each of the following terms in the filter examines various packet header fields and assigns the appropriate forwarding class and packet loss priority:

   - The term **corp-traffic** matches all IPv4 packets with a **10.1.1.0/24** source address and assigns the packets to forwarding class **no-loss** with a loss priority of **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term corp-traffic from source-address 10.1.1.0/24;
user@switch# set term corp-traffic then forwarding-class no-loss
user@switch# set term corp-traffic then loss-priority low
```

- The term **data-traffic** matches all IPv4 packets with a **10.1.2.0/24** source address and assigns the packets to forwarding class **be** (best effort) with a loss priority of **medium-high**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term data-traffic from source-address 10.1.2.0/24;
user@switch# set term data-traffic then forwarding-class be
user@switch# set term data-traffic then loss-priority medium-high
```

- Because the loss of network-generated packets can jeopardize proper network operation, the delay of these packets is preferable to discarding these packets. The term **network-traffic** assigns the packets with an IP precedence of **net-control** to forwarding class **nc** (network control) with a loss priority of **low**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term network-traffic from precedence net-control
user@switch# set term network-traffic then forwarding-class nc
user@switch# set term network-traffic then loss-priority low
```

- The last term **accept-traffic** matches any packets that did not match on any of the preceding terms and assigns the packets to forwarding class **be** with a loss priority of **high**:

```
[edit firewall family ethernet-switching filter ingress-filter]
user@switch# set term accept-traffic then forwarding-class be
user@switch# set term accept-traffic then loss-priority high
```

3. Apply the filter **ingress-filter** to a port, VLAN, or Layer 3 interface. For information about applying the filter, see "Configuring Firewall Filters" on page 1826. (Assigning forwarding classes and PLP is supported only on ingress filters.)

RELATED DOCUMENTATION

Monitoring Firewall Filter Traffic | **827**

Overview of Policers | **2198**

Understanding CoS Classifiers

Understanding CoS Forwarding Classes

Configuring Color-Blind Egress Policers for Medium-Low PLP

If you use color-blind mode and want to configure an egress policer that marks packets to have medium-low PLP, you must configure a single-rate two-color policer at the `[edit firewall policer policer-name]` hierarchy level, because color-blind mode does not support medium-low priority. For example:

1. Specify the name of the policer, the bandwidth limit in bits per second (bps) to control the traffic rate on an interface, and the maximum allowed burst size to control the amount of traffic bursting:

```
[edit]
 user@switch# set firewall policer policer-name if-exceeding bandwidth-limit bytes burst-size-limit bytes
```

2. Specify medium-low loss priority for matching packets:

```
[edit]
 user@switch# set firewall policer policer-name then loss-priority medium-low;
```

3. Apply the filter to a port, VLAN, or Layer 3 interface.

RELATED DOCUMENTATION

- Overview of Policers | 2198
- Understanding Color-Blind Mode for Single-Rate Tricolor Marking | 2222
- Understanding Color-Blind Mode for Two-Rate Tricolor Marking | 2225
- Configuring Firewall Filters | 1826
- Configuring Two-Color and Three-Color Policers to Control Traffic Rates | 2237

Configuring Two-Color and Three-Color Policers to Control Traffic Rates

IN THIS SECTION

- Configuring Two-Color Policers | 2238
- Configuring Three-Color Policers | 2239

Specifying Policers in a Firewall Filter Configuration | 2240
Applying a Firewall Filter That Includes a Policer | 2240

You can rate-limit traffic by configuring a policer and specifying it as an action modifier for a term in a firewall filter. By default, if you specify the same policer in multiple terms, Junos OS creates a separate policer instance for each term and applies rate limiting separately for each instance. For example, if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, each policer instance enforces a 1-Gbps limit. In this case, the total bandwidth allowed by the filter is 3 Gbps.

You can also configure a policer to be filter-specific, which means that Junos OS creates only one policer instance regardless of how many times the policer is referenced. When you do this, rate limiting is applied in aggregate, so if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, the total bandwidth allowed by the filter is 1 Gbps.

NOTE: You can include two-color policer actions on ingress firewall filters only. You can include three-color policer actions on ingress and egress filters.

Configuring Two-Color Policers

To configure a two-color policer:

1. Specify the name of the policer, the bandwidth limit to control the traffic rate on an interface, and the maximum allowed burst size to control the amount of traffic bursting:

```
[edit firewall]
 user@switch# set policer policer-name <filter-specific> if-exceeding bandwidth-limit bps
burst-size-limit bytes
```

   The policer name can contain letters, numbers, and hyphens (-) and can have as many as 64 characters.

   The range for the bandwidth limit is 32000 (32k) through 102,300,000,000 (102300m) bps.

   To determine the value for the burst-size limit, multiply the bandwidth of the interface on which the filter is applied by the amount of time to allow a burst of traffic at that bandwidth to occur and divide the result by 8:

   maximum burst size = (interface bandwidth) X (allowable time for burst) / (8 bits/byte)

The range for the burst-size limit is 1 through 2,147,450,880 bytes.

2. Specify the policer action to discard or assign a loss priority to packets that exceed the rate limits:

```
[edit firewall policer policer-name]
 user@switch# set then (discard | loss-priority low | loss-priority high)
```

Configuring Three-Color Policers

To configure a three-color policer:

1. Specify the name of the policer and (optionally) whether to automatically discard packets with high loss priority (PLP):

```
[edit firewall]
 user@switch# set three-color-policer policer-name
 user@switch# set three-color-policer policer-name action loss-priority high then discard
```

2. Specify whether the three-color policer should be single-rate or two-rate and whether it should be color-aware or color-blind:

```
[edit firewall three-color-policer policer-name]
 user@switch# set (single-rate | two-rate) (color-aware | color-blind)
```

3. For single-rate three-color policers, configure the CIR, CBS, and EBS:

```
[edit firewall three-color-policer policer-name single-rate]
 user@switch# set committed-information-rate bps
 user@switch# set committed-burst-size bytes
 user@switch# set excess-burst-size bytes
```

4. For two-rate three-color policers, configure the CIR, CBS, PIR, and PBS:

```
[edit firewall three-color-policer policer-name single-rate]
 user@switch# set committed-information-rate bps
 user@switch# set committed-burst-size bytes
 user@switch# set peak-information-rate bps
 user@switch# set peak-burst-size bytes
```

Specifying Policers in a Firewall Filter Configuration

To use a two-color policer, configure a filter term that includes the action **policer**:

```
[edit firewall family family-name]
user@switch# set filter filter-name term name then name
```

For example, the following commands apply a two-color policer to all packets sent from 192.0.2.0/24.

```
[edit firewall family family-name]
user@switch# set filter limit-hosts term term1 from source-address 192.0.2.0/24
user@switch# set filter limit-hosts term term1 then policer policer1
```

To use a three-color policer, configure a filter term that includes the action **three-color-policer**:

```
[edit firewall family name]
user@switch# set filter name term name from match-condition
user@switch# set filter name term name then three-color-policer (single-rate | two-rate) name
```

For example, the following commands apply a single-rate three-color policer to all packets received or sent by interface **ge-0/0/6** (depending on whether the filter is an ingress or egress filter).

```
[edit firewall family name]
user@switch# set filter srTCM term term-one from interface ge-0/0/6
user@switch# set filter srTCM term term-one then three-color-policer single-rate srTCM1-ca
```

You must specify whether the three-color policer is single-rate or two-rate, and this must match the policer itself. Otherwise, the configuration listing includes an error message indicating that the three-color policer you referenced in the filter does not exist.

Applying a Firewall Filter That Includes a Policer

A firewall filter that includes one or more policer action modifiers must be applied to a port, VLAN, or Layer 3 interface like any other filter. For information about applying firewall filters, see "Configuring Firewall Filters" on page 1826.

NOTE: You can include two-color policer actions on ingress firewall filters only. You can include three-color policer actions on ingress and egress filters.

RELATED DOCUMENTATION

- Configuring Firewall Filters | 1826
- Overview of Policers | 2198
- Verifying That Two-Color Policers Are Operational | 2241
- Verifying That Three-Color Policers Are Operational | 2242
- Configuring Color-Blind Egress Policers for Medium-Low PLP | 2237

Verifying That Two-Color Policers Are Operational

IN THIS SECTION

- Purpose | 2241
- Action | 2241
- Meaning | 2242

Purpose

Verify that two-color policers in firewall filter configurations are working properly.

Action

Use the `show firewall policer` operational mode command to verify that the policers are working properly:

```
user@switch> show firewall policer
Filter: egress-vlan-watch-employee
Filter: ingress-port-filter
Filter: ingress-port-limit-tcp-icmp
Policers:
Name                                                Packets
icmp-connection-policer                                  10
tcp-connection-policer                                  539
Filter: ingress-vlan-rogue-block
Filter: ingress-vlan-limit-guest
```

Meaning

The `show firewall policer` command displays the names of all firewall filters and policers that are configured. For each policer that is specified in a filter configuration, the output field shows the current packet count for all packets that exceed the specified rate limits.

RELATED DOCUMENTATION

- Configuring Two-Color and Three-Color Policers to Control Traffic Rates | 2237
- Configuring Firewall Filters | 1826
- Monitoring Firewall Filter Traffic | 827

Verifying That Three-Color Policers Are Operational

IN THIS SECTION

- Purpose | 2242
- Action | 2242
- Meaning | 2243

Purpose

Verify that three-color policers in firewall filter configurations are working properly.

Action

Use the following operational mode commands to verify that a three-color policer is working properly:

- `show class-of-service forwarding-table classifiers`
- `show interfaces` *`interface-name`* `extensive`
- `show interfaces queue` *`interface-name`*

Meaning

RELATED DOCUMENTATION

Overview of Policers | 2198

Configuring Two-Color and Three-Color Policers to Control Traffic Rates | 2237

Troubleshooting Policer Configuration

IN THIS SECTION

- Incomplete Count of Packet Drops | 2243
- Counter Reset When Editing Filter | 2244
- Invalid Statistics for Policer | 2245
- Policers Can Limit Egress Filters | 2245

Incomplete Count of Packet Drops

IN THIS SECTION

- Problem | 2243
- Solution | 2244

Problem

Description

Under certain circumstances, Junos OS might display a misleading number of packets dropped by an ingress policer.

If packets are dropped because of ingress admission control, policer statistics might not show the number of packet drops you would expect by calculating the difference between ingress and egress packet counts. This might happen if you apply an ingress policer to multiple interfaces, and the aggregate ingress rate of those interfaces exceeds the line rate of a common egress interface. In this case, packets might be dropped from the ingress buffer. These drops are not included in the count of packets dropped by the policer, which causes policer statistics to underreport the total number of drops.

Solution

This is expected behavior.

Counter Reset When Editing Filter

IN THIS SECTION

- Problem | 2244
- Solution | 2244

Problem

Description

If you edit a firewall filter term, the value of any counter associated with any term in the same filter is set to 0, including the implicit counter for any policer referenced by the filter. Consider the following examples:

- Assume that your filter has `term1`, `term2`, and `term3`, and each term has a counter that has already counted matching packets. If you edit any of the terms in any way, the counters for all the terms are reset to 0.
- Assume that your filter has `term1` and `term2`. Also assume that `term2` has a `policer` action modifier and the implicit counter of the policer has already counted 1000 matching packets. If you edit `term1` or `term2` in any way, the counter for the policer referenced by `term2` is reset to 0.

Solution

This is expected behavior.

Invalid Statistics for Policer

IN THIS SECTION

- Problem | 2245
- Solution | 2245

Problem

Description

If you apply a single-rate two-color policer in more than 128 terms in a firewall filter, the output of the `show firewall` command displays incorrect data for the policer.

Solution

This is expected behavior.

Policers Can Limit Egress Filters

IN THIS SECTION

- Problem | 2245
- Solution | 2246

Problem

Description

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries

for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

Solution

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

SEE ALSO

Overview of Policers | **2198**

Example: Using Policers to Manage Oversubscription | **2232**

Example: Using Filter-Based Forwarding to Route Application Traffic to a Security Device | **1685**

Example: Using Two-Color Policers and Prefix Lists | **2228**

Troubleshooting Policer Configuration

IN THIS SECTION

- Incomplete Count of Packet Drops | 2247
- Counter Reset When Editing Filter | 2248
- Invalid Statistics for Policer | 2248
- Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured | 2249
- Filter-Specific Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured | 2250
- Policers Can Limit Egress Filters | 2251

Incomplete Count of Packet Drops

IN THIS SECTION

- Problem | 2247
- Solution | 2248

Problem

Description

Under certain circumstances, Junos OS might display a misleading number of packets dropped by an ingress policer.

If packets are dropped because of ingress admission control, policer statistics might not show the number of packet drops you would expect by calculating the difference between ingress and egress packet counts. This might happen if you apply an ingress policer to multiple interfaces, and the aggregate ingress rate of those interfaces exceeds the line rate of a common egress interface. In this case, packets might be dropped from the ingress buffer. These drops are not included in the count of packets dropped by the policer, which causes policer statistics to underreport the total number of drops.

Solution

This is expected behavior.

Counter Reset When Editing Filter

IN THIS SECTION

- Problem | 2248
- Solution | 2248

Problem

Description

If you edit a firewall filter term, the value of any counter associated with any term in the same filter is set to 0, including the implicit counter for any policer referenced by the filter. Consider the following examples:

- Assume that your filter has `term1`, `term2`, and `term3`, and each term has a counter that has already counted matching packets. If you edit any of the terms in any way, the counters for all the terms are reset to 0.
- Assume that your filter has `term1` and `term2`. Also assume that `term2` has a `policer` action modifier and the implicit counter of the policer has already counted 1000 matching packets. If you edit `term1` or `term2` in any way, the counter for the policer referenced by `term2` is reset to 0.

Solution

This is expected behavior.

Invalid Statistics for Policer

IN THIS SECTION

- Problem | 2249
- Solution | 2249

Problem

Description

If you apply a single-rate two-color policer in more than 128 terms in a firewall filter, the output of the `show firewall` command displays incorrect data for the policer.

Solution

This is expected behavior.

Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured

IN THIS SECTION

- Problem | **2249**
- Solution | **2250**

Problem

Description

If you configure a policer to rate-limit throughput and apply it on egress to multiple interfaces on a QFX3500 switch or Node, the measured aggregate policed rate might be twice the configured rate, depending on which interfaces you apply the policer to. The doubling of the policed rate occurs if you apply a policer to multiple interfaces and *both* of the following are true:

- There is at least one policed interface in the range xe-0/0/0 to xe-0/0/23 or the range xe-0/1/1 to xe-0/1/7.
- There is at least one policed interface in the range xe-0/0/24 to xe-0/0/47 or the range xe-0/1/8 to xe-0/1/15.

For example, if you configure a policer to rate-limit traffic at 1 Gbps and apply the policer (by using a firewall filter) to xe-0/0/0 and xe-0/0/24 in the output direction, each interface is rate-limited at 1 Gbps, for a total allowed throughput of 2 Gbps. The same behavior occurs if you apply the policer to xe-0/1/1 and xe-0/0/24—each interface is rate-limited at 1 Gbps.

If you apply the same policer on egress to multiple interfaces in these groups, each *group* is rate-limited at 1 Gbps. For example, if you apply the policer to xe-0/0/0 through xe-0/0/4 (five interfaces) and xe-0/0/24 through xe-0/0/33 (ten interfaces), each group is rate-limited at 1 Gbps, for a total allowed throughput of 2 Gbps.

Here is another example: If you apply the policer to xe-0/0/0 through xe-0/0/4 and xe-0/1/1 through xe-0/1/5 (a total of ten interfaces), that group is rate-limited at 1 Gbps in aggregate. If you also apply the policer to xe-0/0/24, that one interface is rate-limited at 1 Gbps while the other ten are still rate-limited at 1 Gbps in aggregate.

Interfaces xe-0/1/1 through xe-0/1/15 are physically located on the QSFP+ uplink ports, according to the following scheme:

- xe-0/1/1 through xe-0/1/3 are on Q0.
- xe-0/1/4 through xe-0/1/7 are on Q1.
- xe-0/1/8 through xe-0/1/11 are on Q2.
- xe-0/1/2 through xe-0/1/15 are on Q3.

The doubling of the policed rate occurs only if the policer is applied in the output direction. If you configure a policer as described above but apply it in the input direction, the total allowed throughput for all interfaces is 1 Gbps.

Solution

This is expected behavior.

Filter-Specific Egress Policers on QFX3500 Devices Might Allow More Throughput Than Is Configured

IN THIS SECTION

- Problem | **2251**
- Solution | **2251**

Problem

Description

You can configure policers to be filter-specific. This means that Junos OS creates only one policer instance no matter how many times the policer is referenced. When you do this, rate limiting is applied in aggregate, so if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms, the total bandwidth allowed by the filter is 1 Gbps. However, the behavior of a filter-specific policer is affected by how the firewall filter terms that reference the policer are stored in ternary content addressable memory (TCAM). If you create a filter-specific policer and reference it in multiple firewall filter terms, the policer allows more traffic than expected if the terms are stored in different TCAM slices. For example, if you configure a policer to discard traffic that exceeds 1 Gbps and reference that policer in three different terms that are stored in three separate memory slices, the total bandwidth allowed by the filter is 3 Gbps, not 1 Gbps.

Solution

To prevent this unexpected behavior, use the information about TCAM slices presented in Planning the Number of Firewall Filters to Create to organize your configuration file so that all the firewall filter terms that reference a given filter-specific policer are stored in the same TCAM slice.

Policers Can Limit Egress Filters

IN THIS SECTION

- Problem | **2251**
- Solution | **2252**

Problem

Description

On some switches, the number of egress policers you configure can affect the total number of allowed egress firewall filters. Every policer has two implicit counters that take up two entries in a 1024-entry TCAM. These are used for counters, including counters that are configured as action modifiers in firewall filter terms. (Policers consume two entries because one is used for green packets and one is used for nongreen packets regardless of policer type.) If the TCAM becomes full, you are unable to commit any more egress firewall filters that have terms with counters. For example, if you configure and commit 512 egress policers (two-color, three-color, or a combination of both policer types), all of the memory entries

for counters get used up. If later in your configuration file you insert additional egress firewall filters with terms that also include counters, *none* of the terms in those filters are committed because there is no available memory space for the counters.

Here are some additional examples:

- Assume that you configure egress filters that include a total of 512 policers and no counters. Later in your configuration file you include another egress filter with 10 terms, 1 of which has a counter action modifier. None of the terms in this filter are committed because there is not enough TCAM space for the counter.
- Assume that you configure egress filters that include a total of 500 policers, so 1000 TCAM entries are occupied. Later in your configuration file you include the following two egress filters:
  - Filter A with 20 terms and 20 counters. All the terms in this filter are committed because there is enough TCAM space for all the counters.
  - Filter B comes after Filter A and has five terms and five counters. *None* of the terms in this filter are committed because there is not enough memory space for *all* the counters. (Five TCAM entries are required but only four are available.)

Solution

You can prevent this problem by ensuring that egress firewall filter terms with counter actions are placed earlier in your configuration file than terms that include policers. In this circumstance, Junos OS commits policers even if there is not enough TCAM space for the implicit counters. For example, assume the following:

- You have 1024 egress firewall filter terms with counter actions.
- Later in your configuration file you have an egress filter with 10 terms. None of the terms have counters but one has a policer action modifier.

You can successfully commit the filter with 10 terms even though there is not enough TCAM space for the implicit counters of the policer. The policer is committed without the counters.

SEE ALSO

Overview of Policers

Example: Using Policers to Manage Oversubscription

Example: Using Two-Color Policers and Prefix Lists

4

PART

Configuration Statements and Operational Commands

Firewall Filter Configuration Statements Supported by Junos OS for EX Series Switches | 2254

gtp-header | 2259

gtp-teid | 2261

per-logical-interface-firewall | 2263

Junos CLI Reference Overview | 2265

Firewall Filter Configuration Statements Supported by Junos OS for EX Series Switches

You configure firewall filters to filter packets based on their components and to perform an action on packets that match the filter.

Table 128 on page 2254 lists the options that are supported for the firewall statement in Junos OS for EX Series switches.

Table 128: Supported Options for Firewall Filter Statements

Statement and Option	Description
`family` *`family-name`* { } 	The ***family-name*** option specifies the version or type of addressing protocol: • **any**—Filter packets based on protocol-independent match conditions. • **ethernet-switching**—Filter Layer 2 (Ethernet) packets and Layer 3 (IP) packets • **inet**—Filter IPv4 packets • **inet6**—Filter IPv6 packets
`filter` *`filter-name`* { } 	The ***filter-name*** option 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 the name in quotation marks (" ").
`interface-specific`	The `interface-specific` statement configures unique names for individual firewall counters specific to each interface.
`term` *`term-name`* { } 	The ***term-name*** option 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 the entire name in quotation marks (" "). Each term name must be unique within a filter.

Table 128: Supported Options for Firewall Filter Statements *(Continued)*

Statement and Option	Description

from {
match-conditions;
}
	The from statement is optional. If you omit it, all packets are considered to match.

then {
action;
action-modifiers;
}
	For information about the ***action*** and ***action-modifiers*** options, see "Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches" on page 1554.

policer *policer-name* { }	The ***policer-name*** option identifies the policer. The name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose the name in quotation marks (" ").
filter-specific	The filter-specific statement configures policers and counters for a specific filter name.

Table 128: Supported Options for Firewall Filter Statements *(Continued)*

Statement and Option	Description

if-exceeding {
bandwidth-limit
bps
burst-size-limit
bytes
}
	The **bandwidth-limit *bps*** option specifies the traffic rate in bits per second (bps).
	
	You can specify ***bps*** as a decimal value or as a decimal number followed by one of the following abbreviations:
	
	• k (thousand)
	• m (million)
	• g (billion, which is also called a thousand million)
	
	Range: 1000 (1k) through 102,300,000,000 (102.3g) bps
	
	The **burst-size-limit *bytes*** option specifies the maximum allowed burst size to control the amount of traffic bursting. To determine the value for the burst-size limit, you can multiply the bandwidth of the interface on which the filter is applied by the amount of time (in seconds) to allow a burst of traffic at that bandwidth to occur:
	
	burst size = bandwidth * allowable time for burst traffic
	
	You can specify a decimal value or a decimal number followed by k (thousand) or m (million).
	
	Range: 1 through 2,147,450,880 bytes

then {
policer-action
}
	Use the ***policer-action*** option to specify **discard** to discard traffic that exceeds the rate limits.

Junos OS for EX Series switches does not support some of the firewall filter statements that are supported by other Junos OS packages. Table 129 on page 2257 shows the firewall filter statements that are not supported by Junos OS for EX Series switches.

Table 129: Firewall Filter Statements That Are Not Supported by Junos OS for EX Series Switches

Statements Not Supported	Statement Hierarchy Level

• interface-set *interface-set-name* {
}
• load-balance-group *group-name* {
}
• three-color-policer *name* {
}
• logical-interface-policer;
• single-rate {
}
• two-rate {
}
	[edit firewall]

Table 129: Firewall Filter Statements That Are Not Supported by Junos OS for EX Series Switches *(Continued)*

Statements Not Supported	Statement Hierarchy Level

• prefix-action *name* {
}
• prefix-policer {
}
• service-filter *filter-name* {
}
• simple-filter *simple-filter-name* {
}
	[edit firewall family *family-name*]

• accounting-profile *name*;	[edit firewall family *family-name* filter *filter-name*]
• logical-bandwidth-policer; • logical-interface-policer;	[edit firewall policer *policer-name*]
bandwidth-percent *number*;	[edit firewall policer *policer-name* if-exceeding]

RELATED DOCUMENTATION

Firewall Filter Match Conditions, Actions, and Action Modifiers for EX Series Switches | 1554

Example: Configuring Firewall Filters for Port, VLAN, and Router Traffic on EX Series Switches | 1650

Configuring Firewall Filters (CLI Procedure) | 1638

Configuring Policers to Control Traffic Rates (CLI Procedure) | 2215

Firewall Filters for EX Series Switches Overview | 1535

gtp-header

IN THIS SECTION

- Syntax | 2259
- Hierarchy Level | 2259
- Description | 2260
- Required Privilege Level | 2260
- Release Information | 2261

Syntax

```
gtp-header
```

Hierarchy Level

```
[edit firewall family family-name filter filter-name from]
```

Description

Include the match conditions for the GTP packet. An example is as below:

```
firewall {
   family inet /inet6 {
           filter gtp_filter {
from {
       protocol udp;
       port 2123;
       gtp-header {
           gtp-teid  <TEID value/list/range>;
          ipv4/ipv6 {
               source-address <32bit for v4 and 128 bit for v6>;
               source-prefix-list <prefix-list>;
               destination-address <32bit for v4 and 128 bit for v6>;
               destination-prefix-list <prefix-list>;
               Protocol/next-header <8 bit>;
               Protocol/next-header-except <8 bit>;
               source-port <16 bit>;
               source-port-except <16 bit>;
               destination-port <16 bit>;
               destination-port-except <16 bit>;
       }
}
then {
      count gtp;
      }
```

Required Privilege Level

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

Release Information

Statement introduced in Junos 22.2R1

gtp-teid

IN THIS SECTION

- Syntax | 2261
- Hierarchy Level | 2261
- Description | 2262
- Options | 2262
- Required Privilege Level | 2262
- Release Information | 2262

Syntax

```
gtp-teid list;
```

Hierarchy Level

```
[edit firewall family family-name filter filter-name from gtp-header]
```

Description

Specify the teid value or list of values in the GTP-C packet header. The firewall filter will attempt to match on the provided teid value or list of values.

Options

gtp-teid *list*

- list - gtp-teid is integer and will take a list of values from 0 to 4294967295. As an example, in the following configuration statement, gtp-teid is supplied a list of two values.

```
set firewall family inet filter ft term 1 from gtp-header gtp-teid [100 200]
```

In the following configuration statement, gtp-teid is supplied a single value.

```
set firewall family inet filter ft term 1 from gtp-header gtp-teid 105
```

Required Privilege Level

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

Release Information

Statement introduced in Junos 22.2R1

per-logical-interface-firewall

IN THIS SECTION

- Syntax | 2263
- Hierarchy Level | 2263
- Description | 2263
- Caveats | 2264
- Required Privilege Level | 2265
- Release Information | 2265

Syntax

```
per-logical-interface-firewall
```

Hierarchy Level

```
[edit chassis]
```

Description

Enables per logical interface firewall filtering in the ingress direction. When enabled, the same set of match conditions and actions that are used for port firewall filters can be used for firewall filters on logical interfaces. The following example depicts the creation of a firewall filter and it being subsequently applied to a logical interface, after the enabling of the `per-logical-interface-firewall` setting.

```
firewall {
    family ethernet-switching {
```

```
        filter <filter-name> {
            term <rule-name> {
                from {
                    source-mac-address {
                        <mac-address>;
                    }
                }
                then {
                    count <count>;
                    policer <policer>;
                }
            }
        }
    }
    policer <policer-name> {
        if-exceeding {
            bandwidth-limit <bandwidth-limit>;
            burst-size-limit <burst-size-limit>;
        }
        then discard;
    }
}
interfaces {
    <interface-name> {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit <interface-unit-number> {
            vlan-id <vlan-id>;
            family ethernet-switching {
                filter {
                    input <filter-name>;
                }
            }
        }
    }
```

Caveats

- `per-logical-interface-firewall` is not supported on enterprise style logical interfaces.

- Per logical interface firewall filtering with mix of services provider and enterprise logical interfaces is not supported.
- `per-logical-interface-firewall` scope is limited to non-VxLAN interfaces.
- With `per-logical-interface-firewall`, IPv6 address in filters across ifls of an ifd should be exclusive.
- Interface specific knob is not recommended with IPv6 address match.
- IFLs belongs to different vlans cannot have the same filter with IPv6 address match.

Required Privilege Level

interface

Release Information

Statement introduced in Junos OS Release 22.2R1 (QFX5110, QFX5120-32C, QFX5120-48T, QFX5120-48Y, QFX5120-48YM, QFX5200, and QFX5210)

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Learn about the syntax and options that make up the statements and commands and understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- *Junos CLI Reference*

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- *Configuration Statements*
- *CLI Commands*

5

PART

Troubleshooting

Knowledge Base | 2267

CHAPTER 35

Knowledge Base