Engineering
Simplicity

Junos® OS

Layer 2 VPNs and VPLS User Guide for Routing Devices

Published
2025-12-19

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

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

Table of Contents

About This Guide | xix

1 **Common Configuration for All VPNs**

VPNs Overview | 2

VPLS | **2**

Types of VPNs | **2**

VPNs and Logical Systems | **6**

Layer 2 VPNs | **7**

Routers in a VPN | **7**

Assigning Routing Instances to VPNs | 9

Configuring Routing Instances on PE Routers in VPNs | **9**

Configuring Virtual-Router Routing Instances in VPNs | **15**

Configuring Path MTU Checks for VPN Routing Instances | **18**

Distributing Routes in VPNs | 20

Enabling Routing Information Exchange for VPNs | **20**

Configuring IBGP Sessions Between PE Routers in VPNs | **20**

Configuring Aggregate Labels for VPNs | **22**

Configuring a Signaling Protocol and LSPs for VPNs | **23**

Configuring Policies for the VRF Table on PE Routers in VPNs | **28**

Configuring the Route Origin for VPNs | **36**

Distributing VPN Routes with Target Filtering | 41

Configuring BGP Route Target Filtering for VPNs | **41**

Example: BGP Route Target Filtering for VPNs | **43**

Example: Configuring BGP Route Target Filtering for VPNs | **46**

Configuring Static Route Target Filtering for VPNs | **58**

Understanding Proxy BGP Route Target Filtering for VPNs | 59

Example: Configuring Proxy BGP Route Target Filtering for VPNs | 60

- Requirements | 60
- Overview | 61
- Configuration | 62
- Verification | 81

Example: Configuring an Export Policy for BGP Route Target Filtering for VPNs | 82

- Requirements | 83
- Overview | 83
- Configuration | 85
- Verification | 105

Reducing Network Resource Use with Static Route Target Filtering for VPNs | 106

Configuring Forwarding Options for VPNs | 108

Chained Composite Next Hops for VPNs and Layer 2 Circuits | 108

Example: Configuring Chained Composite Next Hops for Direct PE-PE Connections in VPNs | 109

- Requirements | 110
- Overview and Topology | 110
- Configuration | 110

Configuring Graceful Restart for VPNs | 118

VPN Graceful Restart | 118

Configuring Graceful Restart for VPNs | 119

Enabling Unicast Reverse-Path Forwarding Check for VPNs | 122

Understanding and Preventing Unknown Unicast Forwarding | 123

- Verifying That Unknown Unicast Packets Are Forwarded to a Single Interface | 124
- Configuring Unknown Unicast Forwarding (ELS) | 125
  - Configuring Unknown Unicast Forwarding on EX4300 Switches | 125
  - Configuring Unknown Unicast Forwarding on EX9200 Switches | 126
- Verifying That Unknown Unicast Packets Are Forwarded to a Trunk Interface | 128
- Configuring Unknown Unicast Forwarding (CLI Procedure) | 130
- Configuring Unknown Unicast Drop (CLI Procedure) | 130
- Platform-Specific Configuring Unknown Unicast Forwarding Behavior | 132

Configuring Class of Service for VPNs | 133

VPNs and Class of Service | **133**

Rewriting Class of Service Markers and VPNs | **133**

Pinging VPNs | 134

Pinging VPNs, VPLS, and Layer 2 Circuits | **134**

Setting the Forwarding Class of the Ping Packets | **135**

Pinging a VPLS Routing Instance | **135**

Pinging a Layer 2 VPN | **136**

Pinging a Layer 3 VPN | **137**

Pinging a Layer 2 Circuit | **137**

Pinging Customer Edge Device IP Address | **137**

2 **Common Configuration for Layer 2 VPNs and VPLS**

Overview | 144

Understanding Layer 2 VPNs | **144**

Layer 2 VPN Applications | **145**

Supported Layer 2 VPN Standards | **146**

Layer 2 VPNs Configuration Overview | 148

Introduction to Configuring Layer 2 VPNs | **148**

Configuring the Local Site on PE Routers in Layer 2 VPNs | **150**

Layer 2 VPN Configuration Example | **157**

Example: Configure MPLS-Based Layer 2 VPNs | **180**

- Requirements | **181**
- Overview and Topology | **182**
- Quick Configurations | **183**
- Configure the Local PE (PE1) Device for a MPLS-Based Layer 2 VPN | **186**
- Configure the Remote PE (PE2) Device for a MPLS-Based Layer 2 VPN | **194**
- Verification | **200**

Transmitting Nonstandard BPDUs in Layer 2 VPNs and VPLS | **209**

Configuring Layer 2 Interfaces | 210

Configuring CCC Encapsulation for Layer 2 VPNs | **210**

Configuring TCC Encapsulation for Layer 2 VPNs and Layer 2 Circuits | **211**

Configuring the MTU for Layer 2 Interfaces | **213**

Disabling the Control Word for Layer 2 VPNs | **214**

Configuring Path Selection for Layer 2 VPNs and VPLS | 216

Understanding BGP Path Selection | **216**

Enabling BGP Path Selection for Layer 2 VPNs and VPLS | **221**

Creating Backup Connections with Redundant Pseudowires | 224

Redundant Pseudowires for Layer 2 Circuits and VPLS | **224**

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | **226**

Configuring Class of Service for Layer 2 VPNs | 230

Configuring Traffic Policing in Layer 2 VPNs | **230**

Monitoring Layer 2 VPNs | 232

Configuring BFD for Layer 2 VPN and VPLS | **232**

BFD Support for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS | **234**

Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS | **235**

Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview | **237**

Configure a MEP to Generate and Respond to CFM Protocol Messages | **239**

- Configure a Maintenance Association End Point (MEP) | **240**
- Configure a Remote Maintenance Association End Point (MEP) | **242**

3 **Configuring Group VPNs**

Configuring Group VPNv2 | 245

Group VPNv2 Overview | **245**

Configuring Group VPNs in Group VPNv2 on Routing Devices | **269**

Group VPN on AMS interfaces | **272**

Use Case for Configuring Group VPNv2 | **273**

Example: Configuring Group VPNs in Group VPNv2 on Routing Devices | **274**

- Requirements | **275**
- Overview | **275**
- Configuration | **277**
- Verification | **289**
- Troubleshooting | **292**

4 **Configuring Public Key Infrastructure**

Configuring Digital Certificate Validation | 298

Validate Digital Certificate on MX Series Devices | **298**

Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device | **305**

- Requirements | **305**
- Overview | **305**
- Configuration | **306**
- Verification | **307**

Configuring a Device for Certificate Chains | 310

IKE Authentication (Certificate-Based Authentication) | **310**

Example: Configuring a Device for Peer Certificate Chain Validation | **313**

- Requirements | **313**
- Overview | **314**
- Configuration | **315**
- Verification | **322**
- IKE and IPsec SA Failure for a Revoked Certificate | **324**

Managing Certificate Revocation | 326

Online Certificate Status Protocol and Certificate Revocation Lists | **326**

Example: Improving Security by Configuring OCSP for Certificate Revocation Status | **329**

- Requirements | **329**
- Overview | **329**
- Configuration | **332**
- Verification | **343**

5 **Configuring Layer 2 Circuits**

Overview | 351

Layer 2 Circuit Overview | **351**

Layer 2 Circuits Configuration Overview | 353

Configuring Static Layer 2 Circuits | **353**

Configuring Local Interface Switching in Layer 2 Circuits | **354**

Configuring Interfaces for Layer 2 Circuits | **357**

Configuring Aggregated Ethernet Interfaces for Layer 2 Circuits | **367**

Example: Configuring the Pseudowire Status TLV | **369**

- Requirements | **369**
- Overview | **369**
- Configuration | **369**

Configuring Policies for Layer 2 Circuits | **371**

Configuring LDP for Layer 2 Circuits | **376**

Configuring Class of Service with Layer 2 Circuits | 377

Layer 2 Circuit Bandwidth Accounting and Call Admission Control | **377**

Configuring Bandwidth Allocation and Call Admission Control in Layer 2 Circuits | **380**

Configuring Pseudowire Redundancy for Layer 2 Circuits | 382

Understanding Pseudowire Redundancy Mobile Backhaul Scenarios | **382**

Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario | **388**

- Requirements | **388**
- Overview | **388**
- Configuration | **389**
- Verification | **414**

Extension of Pseudowire Redundancy Condition Logic to Pseudowire Service Logical Interface Overview | **419**

Configuring Load Balancing for Layer 2 Circuits | 423

Configuring Protection Features for Layer 2 Circuits | 424

Egress Protection LSPs for Layer 2 Circuits | **424**

Configuring Egress Protection Service Mirroring for BGP Signaled Layer 2 Services | **426**

Example: Configuring an Egress Protection LSP for a Layer 2 Circuit | 431

- Requirements | 431
- Egress Protection LSP Overview | 432
- Egress Protection LSP Configuration | 434

Example: Configuring Layer 2 Circuit Protect Interfaces | 447

Example: Configuring Layer 2 Circuit Switching Protection | 454

- Requirements | 454
- Overview | 455
- Configuration | 456

Monitoring Layer 2 Circuits with BFD | 474

Configuring BFD for VCCV for Layer 2 Circuits | 474

Example: Configuring BFD for VCCV for Layer 2 Circuits | 477

- Requirements | 477
- Overview | 478
- Configuration | 479
- Verification | 485

Troubleshooting Layer 2 Circuits | 489

Tracing Layer 2 Circuit Operations | 489

6 **Configuring VPWS VPNs**

Overview | 491

Understanding VPWS | 491

Supported VPWS Standards | 495

FAT Flow Labels Overview | 496

Configuring VPWS VPNs | 498

Understanding FEC 129 BGP Autodiscovery for VPWS | 498

Example: Configuring FEC 129 BGP Autodiscovery for VPWS | 501

- Requirements | 502
- Overview | 502
- Configuration | 507
- Verification | 513

Example: Configuring MPLS Egress Protection Service Mirroring for BGP Signaled Layer 2 Services | **519**

Requirements | **519**

Overview | **519**

Configuration | **521**

Verification | **538**

Understanding Multisegment Pseudowire for FEC 129 | **543**

Example: Configuring a Multisegment Pseudowire | **548**

Requirements | **549**

Overview | **549**

Configuration | **556**

Verification | **584**

Troubleshooting | **595**

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | **598**

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | **601**

7

Configuring VPLS

Overview | 606

Introduction to VPLS | **606**

Supported VPLS Standards | **607**

Supported Platforms and PICs | **607**

VPLS Configuration Overview | 609

Introduction to Configuring VPLS | **609**

Configuring an Ethernet Switch as the CE Device for VPLS | **621**

Configuring Signaling Protocols for VPLS | 622

VPLS Routing and Virtual Ports | **622**

BGP Signaling for VPLS PE Routers Overview | **624**

Control Word for BGP VPLS Overview | **625**

Configuring a Control Word for BGP VPLS | **626**

BGP Route Reflectors for VPLS | **628**

Interoperability Between BGP Signaling and LDP Signaling in VPLS | **629**

Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS | **632**

Example: VPLS Configuration (BGP Signaling) | **637**

Example: VPLS Configuration (BGP and LDP Interworking) | **653**

Assigning Routing Instances to VPLS | 675

Configuring VPLS Routing Instances | **675**

Configuring a VPLS Routing Instance | **693**

Support of Inner VLAN List and Inner VLAN Range for Qualified BUM Pruning on a Dual-Tagged Interface for a VPLS Routing Instance Overview | **694**

Configuring Qualified BUM Pruning for a Dual-Tagged Interface with Inner VLAN list and InnerVLAN range for a VPLS Routing Instance | **696**

Configuring a Layer 2 Control Protocol Routing Instance | **698**

PE Router Mesh Groups for VPLS Routing Instances | **699**

Configuring VPLS Fast Reroute Priority | **700**

Specifying the VT Interfaces Used by VPLS Routing Instances | **702**

Understanding PIM Snooping for VPLS | **703**

Example: Configuring PIM Snooping for VPLS | **704**

- Requirements | **704**
- Overview | **704**
- Configuration | **706**
- Verification | **715**

VPLS Label Blocks Operation | **719**

Configuring the Label Block Size for VPLS | **724**

Example: Building a VPLS From Router 1 to Router 3 to Validate Label Blocks | **725**

- Requirements | **726**
- Overview and Topology | **726**
- Configuration | **727**

Associating Interfaces with VPLS | 734

Configuring Interfaces for VPLS Routing | **734**

- Configuring the VPLS Interface Name | **735**

Configuring VPLS Interface Encapsulation | **736**

Enabling VLAN Tagging | **738**

Configuring VLAN IDs for Logical Interfaces | **739**

Enabling VLANs for Hub and Spoke VPLS Networks | **740**

Sample Scenario of Hierarchical Virtual Private LAN Service on Logical Tunnel Interface | **740**

Configuring Aggregated Ethernet Interfaces for VPLS | **741**

VPLS and Aggregated Ethernet Interfaces | **743**

Configuring VLAN Identifiers for VLANs and VPLS Routing Instances | **744**

Enabling VLAN Tagging | **749**

Configuring VPLS Without a Tunnel Services PIC | **749**

Configuring Pseudowires | 751

Configuring Static Pseudowires for VPLS | **751**

VPLS Path Selection Process for PE Routers | **753**

BGP and VPLS Path Selection for Multihomed PE Routers | **755**

Dynamic Profiles for VPLS Pseudowires | **757**

Use Cases for Dynamic Profiles for VPLS Pseudowires | **758**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Basic Solutions | **759**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Complex Solutions | **764**

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | **770**

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | **772**

Example: Configuring H-VPLS BGP-Based and LDP-Based VPLS Interoperation | **774**

Requirements | **775**

Overview | **775**

Configuration | **776**

Verification | **793**

Example: Configuring BGP-Based H-VPLS Using Different Mesh Groups for Each Spoke Router | **805**

Requirements | **805**

Overview and Topology | **806**

Configuration | **808**

Example: Configuring LDP-Based H-VPLS Using a Single Mesh Group to Terminate the Layer 2 Circuits | **833**

- Requirements | **834**
- Overview and Topology | **834**
- Configuration | **835**

Example: Configuring H-VPLS With VLANs | **842**

- Requirements | **842**
- Overview | **842**
- Configuration | **844**
- Verification | **855**

Example: Configuring H-VPLS Without VLANs | **860**

- Requirements | **860**
- Overview | **860**
- Configuration | **861**
- Verification | **871**

Configure Hot-Standby Pseudowire Redundancy in H-VPLS | **876**

Sample Scenario of H-VPLS on ACX Series Routers for IPTV Services | **881**

Configuring Multihoming | 885

VPLS Multihoming Overview | **885**

Advantages of Using Autodiscovery for VPLS Multihoming | **888**

Example: Configuring FEC 129 BGP Autodiscovery for VPWS | **889**

- Understanding VPWS | **889**
- Understanding FEC 129 BGP Autodiscovery for VPWS | **892**
- Example: Configuring FEC 129 BGP Autodiscovery for VPWS | **895**
  - Requirements | **896**
  - Overview | **896**
  - Configuration | **901**
  - Verification | **907**

Example: Configuring BGP Autodiscovery for LDP VPLS | **913**

- Requirements | **913**
- Overview | **914**
- Configuration | **916**

Verification | 937

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938

Requirements | 938

Overview | 938

Configuration | 939

Verification | 947

VPLS Multihoming Reactions to Network Failures | 953

Configuring VPLS Multihoming | 954

Example: VPLS Multihoming, Improved Convergence Time | 957

Requirements | 958

Overview | 958

Configuration | 960

Example: Configuring VPLS Multihoming (FEC 129) | 974

VPLS Multihoming Overview | 975

Example: Configuring VPLS Multihoming (FEC 129) | 977

Requirements | 977

Overview | 977

Configuration | 979

Verification | 989

Next-Generation VPLS for Multicast with Multihoming Overview | 994

Example: Next-Generation VPLS for Multicast with Multihoming | 1000

Requirements | 1000

Overview and Topology | 1001

Configuration | 1004

Configuring Point-to-Multipoint LSPs | 1028

Next-Generation VPLS Point-to-Multipoint Forwarding Overview | 1028

Example: NG-VPLS Using Point-to-Multipoint LSPs | 1034

Requirements | 1035

Overview and Topology | 1035

Configuration | 1038

Flooding Unknown Traffic Using Point-to-Multipoint LSPs in VPLS | 1076

Example: Configuring Ingress Replication for IP Multicast Using MBGP MVPNs | **1081**

- Requirements | **1081**
- Overview | **1081**
- Configuration | **1084**
- Verification | **1089**

Mapping VPLS Traffic to Specific LSPs | **1099**

Configuring Inter-AS VPLS and IRB VPLS | 1101

Example: Configuring Inter-AS VPLS with MAC Processing at the ASBR | **1101**

- Requirements | **1101**
- Overview and Topology | **1102**
- Configuration | **1103**

Configuring VPLS and Integrated Routing and Bridging | **1137**

Configuring Integrated Routing and Bridging in a VPLS Instance (MX Series Routers Only) | **1141**

Configuring L2PT for VPLS | **1142**

Configuring Load Balancing and Performance | 1145

Configuring VPLS Load Balancing | **1146**

Configuring VPLS Load Balancing Based on IP and MPLS Information | **1148**

Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms | **1150**

Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves | **1152**

- MAC Moves Loop Prevention in VPLS Network Overview | **1152**
- Configuring VPLS Loop Prevention Due to MAC Moves | **1154**
- Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves | **1156**
  - Requirements | **1156**
  - Overview | **1157**
  - Configuration | **1158**
  - Verification | **1168**

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**

Configuring MAC Pinning on VPLS CE Interface | **1185**

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance | **1187**

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance | **1189**

Configuring MAC Pinning on Access Interfaces for Logical Systems | **1191**

Configuring MAC Pinning on Trunk Interfaces for Logical Systems | **1193**

Configuring MAC Pinning on Access Interfaces in Virtual Switches for Logical Systems | **1195**

Configuring MAC Pinning on Trunk Interfaces in Virtual Switches for Logical Systems | **1198**

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) for Logical Systems | **1200**

Configuring MAC Pinning on VPLS CE Interface for Logical Systems | **1203**

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance for Logical Systems | **1205**

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance for Logical Systems | **1207**

Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Access Interfaces | **1209**

- Requirements | **1210**
- Overview | **1210**
- Configuration | **1211**
- Verification | **1214**

Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Trunk Interfaces | **1215**

- Requirements | **1216**
- Overview | **1216**
- Configuration | **1217**
- Verification | **1223**

Configuring Improved VPLS MAC Address Learning on T4000 Routers with Type 5 FPCs | **1225**

Understanding Qualified MAC Learning | **1227**

Qualified Learning VPLS Routing Instance Behavior | **1228**

Configuring Qualified MAC Learning | **1235**

Configuring Class of Service and Firewall Filters in VPLS | 1237

Configuring EXP-Based Traffic Classification for VPLS | **1237**

Configuring Firewall Filters and Policers for VPLS | **1238**

Firewall Filter Match Conditions for VPLS Traffic | **1243**

Monitoring and Tracing VPLS | 1263

Configuring Port Mirroring for VPLS Traffic | **1263**

Configuring Y.1731 Functionality for VPLS to Support Delay and Delay Variation | **1263**

Tracing VPLS Traffic and Operations | **1265**

8 **Connecting Layer 2 VPNs and Circuits to Other VPNs**

Connecting Layer 2 VPNs to Other VPNs | 1267

Layer 2 VPN to Layer 2 VPN Connections | **1267**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 VPN to a Layer 2 VPN | **1268**

Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN | **1270**

- Requirements | **1270**
- Overview and Topology | **1271**
- Configuration | **1272**

Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview | **1292**

Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN | **1294**

- Requirements | **1294**
- Overview and Topology | **1294**
- Configuration | **1298**
- Verification | **1318**

Configure an Anycast Gateway in an L2VPN | **1325**

Stitching Layer 2 VPNs and VPWS | **1328**

Connecting Layer 2 Circuits to Other VPNs | 1333

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 Circuit to a Layer 2 VPN | **1333**

Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN | **1335**

- Requirements | **1335**
- Overview and Topology | **1335**
- Configuration | **1337**

Example: Interconnecting a Layer 2 Circuit with a Layer 2 Circuit | **1346**

- Requirements | **1347**
- Overview and Topology | **1347**
- Configuration | **1349**

Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN | **1368**

Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN | **1369**

- Requirements | **1369**
- Overview and Topology | **1370**
- Configuration | **1371**
- Verifying the Layer 2 Circuit to Layer 3 VPN Interconnection | **1385**

9 **Configuration Statements and Operational Commands**

Junos CLI Reference Overview | 1397

About This Guide

The Junos operating system (Junos OS) supports layer 2 VPN service which allows customers to have geographically dispersed private networks across service provider's networks. Use the topics on this page to configure VPWS, VPLS, and layer 2 VPN routing instances to enable layer 2 VPN service.

RELATED DOCUMENTATION

Learn About Secure VPNs

Day One Poster: VPNs

FAQ: VPLS on MX Series Routers

FAQ: Layer 2 Circuits and Layer 2 VPN on MX Series Routers

1

PART

Common Configuration for All VPNs

- VPNs Overview | **2**
- Assigning Routing Instances to VPNs | **9**
- Distributing Routes in VPNs | **20**
- Distributing VPN Routes with Target Filtering | **41**
- Configuring Forwarding Options for VPNs | **108**
- Configuring Graceful Restart for VPNs | **118**
- Configuring Class of Service for VPNs | **133**
- Pinging VPNs | **134**

CHAPTER 1

VPNs Overview

IN THIS CHAPTER

- VPLS | 2
- Types of VPNs | 2
- VPNs and Logical Systems | 6
- Layer 2 VPNs | 7
- Routers in a VPN | 7

VPLS

In a Layer 3 network only, you can configure virtual private LAN service (VPLS), which is an Ethernet-based point-to-multipoint Layer 2 VPN. It enables you to connect geographically dispersed Ethernet local area networks (LAN) sites to each other across an MPLS backbone. For ISP customers who implement VPLS, all sites appear to be in the same Ethernet LAN even though traffic travels across the service provider's network.

RELATED DOCUMENTATION

Junos OS VPNs Library for Routing Devices

MX Series Router Architecture

Types of VPNs

IN THIS SECTION

- Layer 2 VPNs | 3

Layer 3 VPNs | 4
VPLS | 4
Virtual-Router Routing Instances | 5

A virtual private network (VPN) consists of two topological areas: the provider's network and the customer's network. The customer's network is commonly located at multiple physical sites and is also private (non-Internet). A customer site would typically consist of a group of routers or other networking equipment located at a single physical location. The provider's network, which runs across the public Internet infrastructure, consists of routers that provide VPN services to a customer's network as well as routers that provide other services. The provider's network connects the various customer sites in what appears to the customer and the provider to be a private network.

To ensure that VPNs remain private and isolated from other VPNs and from the public Internet, the provider's network maintains policies that keep routing information from different VPNs separate. A provider can service multiple VPNs as long as its policies keep routes from different VPNs separate. Similarly, a customer site can belong to multiple VPNs as long as it keeps routes from the different VPNs separate.

The Junos® Operating System (Junos OS) provides several types of VPNs; you can choose the best solution for your network environment. Each of the following VPNs has different capabilities and requires different types of configuration:

Layer 2 VPNs

Implementing a Layer 2 VPN on a router is similar to implementing a VPN using other Layer 2 technologies. However, for a Layer 2 VPN on a router, traffic is forwarded to the router in Layer 2 format. It is carried by MPLS over the service provider's network and then converted back to a Layer 2 format at the receiving site. You can configure different Layer 2 formats at the sending and receiving sites.

On a Layer 2 VPN, routing occurs on the customer's routers, typically on the CE router. The CE router connected to a service provider on a Layer 2 VPN must select the appropriate circuit on which to send traffic. The PE router receiving the traffic sends it across the service provider's network to the PE router connected to the receiving site. The PE routers do not need to store or process the customer's routes; they only need to be configured to send data to the appropriate tunnel.

For a Layer 2 VPN, customers need to configure their own routers to carry all Layer 3 traffic. The service provider needs to know only how much traffic the Layer 2 VPN needs to carry. The service provider's routers carry traffic between the customer's sites using Layer 2 VPN interfaces. The VPN topology is determined by policies configured on the PE routers.

Layer 3 VPNs

In a Layer 3 VPN, the routing occurs on the service provider’s routers. Therefore, Layer 3 VPNs require more configuration on the part of the service provider, because the service provider’s PE routers must store and process the customer’s routes.

In the Junos OS, Layer 3 VPNs are based on RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*. This RFC defines a mechanism by which service providers can use their IP backbones to provide Layer 3 VPN services to their customers. The sites that make up a Layer 3 VPN are connected over a provider’s existing public Internet backbone.

VPNs based on RFC 4364 are also known as BGP/MPLS VPNs because BGP is used to distribute VPN routing information across the provider’s backbone, and MPLS is used to forward VPN traffic across the backbone to remote VPN sites.

Customer networks, because they are private, can use either public addresses or private addresses, as defined in RFC 1918, *Address Allocation for Private Internets*. When customer networks that use private addresses connect to the public Internet infrastructure, the private addresses might overlap with the private addresses used by other network users. BGP/MPLS VPNs solve this problem by prefixing a VPN identifier to each address from a particular VPN site, thereby creating an address that is unique both within the VPN and within the public Internet. In addition, each VPN has its own VPN-specific routing table that contains the routing information for that VPN only.

VPLS

Virtual private LAN service (VPLS) allows you to connect geographically dispersed customer sites as if they were connected to the same LAN. In many ways, it works like a Layer 2 VPN. VPLS and Layer 2 VPNs use the same network topology and function similarly. A packet originating within a customer’s network is sent first to a CE device. It is then sent to a PE router within the service provider’s network. The packet traverses the service provider’s network over an MPLS LSP. It arrives at the egress PE router, which then forwards the traffic to the CE device at the destination customer site.

The key difference in VPLS is that packets can traverse the service provider’s network in a point-to-multipoint fashion, meaning that a packet originating from a CE device can be broadcast to PE routers in the VPLS. In contrast, a Layer 2 VPN forwards packets in a point-to-point fashion only. The destination of a packet received from a CE device by a PE router must be known for the Layer 2 VPN to function properly.

In a Layer 3 network only, you can configure virtual private LAN service (VPLS), to connect geographically dispersed Ethernet local area networks (LAN) sites to each other across an MPLS backbone. For ISP customers who implement VPLS, all sites appear to be in the same Ethernet LAN even though traffic travels across the service provider's network. VPLS is designed to carry Ethernet traffic across an MPLS-enabled service provider network. In certain ways, VPLS mimics the behavior of an Ethernet network. When a PE router configured with a VPLS routing instance receives a packet from a CE device, it first checks the appropriate routing table for the destination of the VPLS packet. If the router has the destination, it forwards it to the appropriate PE router. If it does not have the destination, it broadcasts the packet to all the other PE routers that are members of the same VPLS routing instance. The PE routers forward the packet to their CE devices. The CE device that is the intended recipient of the packet forwards it to its final destination. The other CE devices discard it.

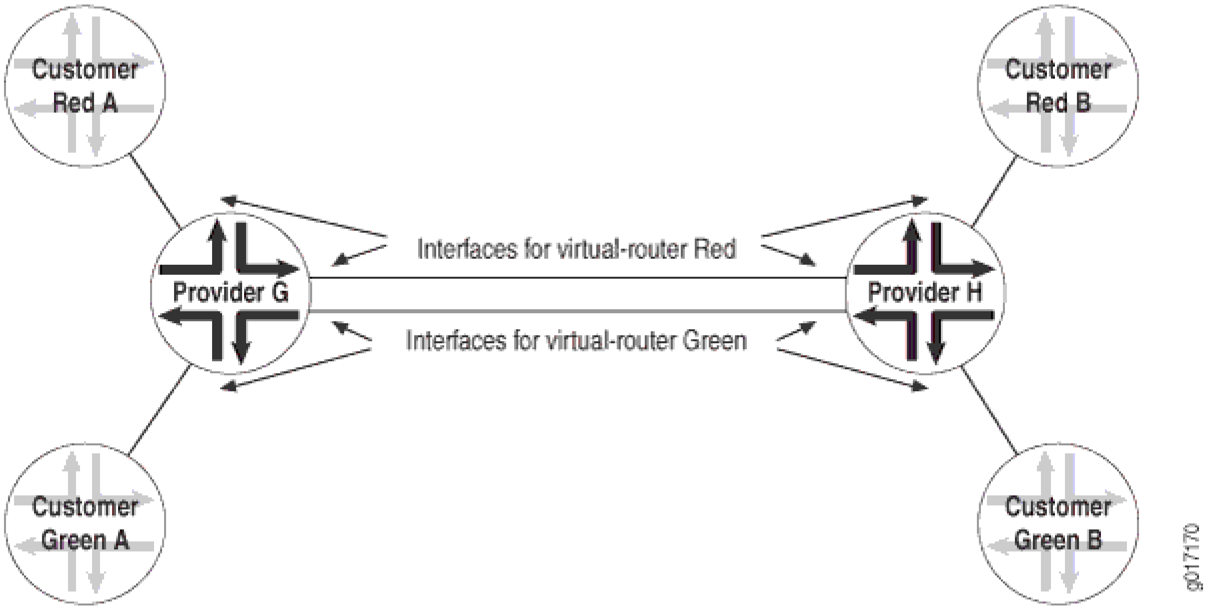
Virtual-Router Routing Instances

A virtual-router routing instance, like a VPN routing and forwarding (VRF) routing instance, maintains separate routing and forwarding tables for each instance. However, many configuration steps required for VRF routing instances are not required for virtual-router routing instances. Specifically, you do not need to configure a route distinguisher, a routing table policy (the `vrf-export`, `vrf-import`, and `route-distinguisher` statements), or MPLS between the P routers.

However, you need to configure separate logical interfaces between each of the service provider routers participating in a virtual-router routing instance. You also need to configure separate logical interfaces between the service provider routers and the customer routers participating in each routing instance. Each virtual-router instance requires its own unique set of logical interfaces to all participating routers.

Figure 1 on page 6 shows how this works. The service provider routers G and H are configured for virtual-router routing instances Red and Green. Each service provider router is directly connected to two local customer routers, one in each routing instance. The service provider routers are also connected to each other over the service provider network. These routers need four logical interfaces: a *logical interface* to each of the locally connected customer routers and a logical interface to carry traffic between the two service provider routers for each virtual-router instance.

Figure 1: Logical Interface per Router in a Virtual-Router Routing Instance



Layer 3 VPNs do not have this configuration requirement. If you configure several Layer 3 VPN routing instances on a PE router, all the instances can use the same logical interface to reach another PE router. This is possible because Layer 3 VPNs use MPLS (VPN) labels that differentiate traffic going to and from various routing instances. Without MPLS and VPN labels, as in a virtual-router routing instance, you need separate logical interfaces to separate traffic from different instances.

One method of providing this logical interface between the service provider routers is by configuring tunnels between them. You can configure IP Security (IPsec), generic routing encapsulation (GRE), or IP-IP tunnels between the service provider routers, terminating the tunnels at the virtual-router instance.

VPNs and Logical Systems

You can partition a single physical router into multiple logical systems that perform independent routing tasks. Because logical systems perform a subset of the tasks once handled by the physical router, logical systems offer an effective way to maximize the use of a single routing platform.

Logical systems perform a subset of the actions of a physical router and have their own unique routing tables, interfaces, policies, and routing instances. A set of logical systems within a single router can handle the functions previously performed by several small routers.

Logical systems support Layer 2 VPNs, Layer 3 VPNs, VPLS, and Layer 2 circuits. For more information about logical systems, see the Logical Systems User Guide for Routers and Switches.

Ethernet VPN (EVPN) support has also been extended to logical systems running on MX devices. The same EVPN options and performance are available, and can be configured under the `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols evpn]` hierarchy.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
17.4	Starting in Junos OS release 17.4R1, Ethernet VPN (EVPN) support has also been extended to logical systems running on MX devices. The same EVPN options and performance are available, and can be configured under the `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols evpn]` hierarchy.

Layer 2 VPNs

In a Layer 3 network only, you can configure Layer 2 virtual private network (VPN) under a Layer 2 VPN routing instance type **l2vpn**.

In a Layer 2 environment, you can use a **l2vpn** routing instance to transparently carry Layer 2 traffic over an IP/MPLS backbone. Layer 2 traffic is sent to the provider edge (PE) router in Layer 2 format. The PE router encapsulates the frames and transports them over the IP/MPLS backbone to the PE router on the other side of the cloud. The remote PE router removes encapsulation and sends the frames to the receiving site in Layer 2 format.

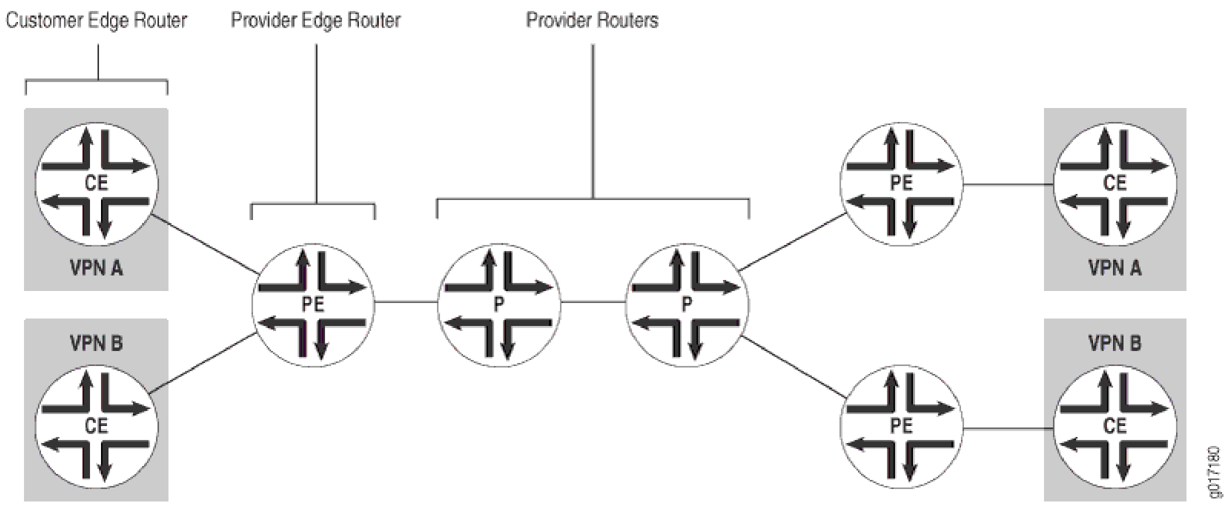
RELATED DOCUMENTATION

- MX Series Router Architecture
- Layer 2 and Layer 3 Features on MX Series Routers
- Junos OS VPNs Library for Routing Devices

Routers in a VPN

Figure 2 on page 8 illustrates how VPN functionality is provided by the provider edge (PE) routers; the provider and customer edge (CE) routers have no special configuration requirements for VPNs.

Figure 2: Routers in a VPN



CHAPTER 2

Assigning Routing Instances to VPNs

IN THIS CHAPTER

- Configuring Routing Instances on PE Routers in VPNs | **9**
- Configuring Virtual-Router Routing Instances in VPNs | **15**
- Configuring Path MTU Checks for VPN Routing Instances | **18**

Configuring Routing Instances on PE Routers in VPNs

IN THIS SECTION

- Configuring the Routing Instance Name for a VPN | **10**
- Configuring the Description | **10**
- Configuring the Instance Type | **11**
- Configuring Interfaces for VPN Routing | **12**
- Configuring the Route Distinguisher | **14**
- Configuring Automatic Route Distinguishers | **15**

You need to configure a routing instance for each VPN on each of the PE routers participating in the VPN. The configuration procedures outlined in this section are applicable to Layer 2 VPNs, Layer 3 VPNs, and VPLS. The configuration procedures specific to each type of VPN are described in the corresponding sections in the other configuration chapters.

To configure routing instances for VPNs, include the following statements:

```
description text;
instance-type type;
interface interface-name;
```

```
route-distinguisher (as-number:number | ip-address:number);
vrf-import [ policy-names ];
vrf-export [ policy-names ];
vrf-target {
    export community-name;
    import community-name;
}
```

You can include these statements at the following hierarchy levels:

- `[edit routing-instances routing-instance-name]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name]`

To configure VPN routing instances, you perform the steps in the following sections:

Configuring the Routing Instance Name for a VPN

The name of the routing instance for a VPN can be a maximum of 128 characters and can contain letters, numbers, and hyphens. In modern Junos OS releases, you cannot specify `default` as the actual routing-instance name. You also cannot use any special characters (! @ # $ % ^ & * , +< > : ;) within the name of a routing instance.

NOTE: You can include a slash (/) in a routing instance name only if a logical system is not configured. That is, you cannot include the slash character in a routing instance name if a logical system other than the default is explicitly configured.

Specify the routing-instance name with the `routing-instance` statement:

```
routing-instance routing-instance-name {...}
```

You can include this statement at the following hierarchy levels:

- `[edit]`
- `[edit logical-systems logical-system-name]`

Configuring the Description

To provide a text description for the routing instance, include the `description` statement. If the text includes one or more spaces, enclose them in quotation marks (" "). Any descriptive text you include is displayed in the output of the `show route instance detail` command and has no effect on the operation of the routing instance.

To configure a text description, include the `description` statement:

```
description text;
```

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]`

Configuring the Instance Type

The instance type you configure varies depending on whether you are configuring Layer 2 VPNs, Layer 3 VPNs, VPLS, or virtual routers. Specify the instance type by including the `instance-type` statement:

- To enable Layer 2 VPN routing on a PE router, include the `instance-type` statement and specify the value `l2vpn`:

  ```
  instance-type l2vpn;
  ```

- To enable VPLS routing on a PE router, include the `instance-type` statement and specify the value `vpls`:

  ```
  instance-type vpls;
  ```

- Layer 3 VPNs require that each PE router have a VPN routing and forwarding (VRF) table for distributing routes within the VPN. To create the VRF table on the PE router, include the `instance-type` statement and specify the value `vrf`:

  ```
  instance-type vrf;
  ```

  NOTE: Routing Engine based sampling is not supported on VRF routing instances.

- To enable the virtual-router routing instance, include the `instance-type` statement and specify the value `virtual-router`:

  ```
  instance-type virtual-router;
  ```

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`*`]`

Configuring Interfaces for VPN Routing

On each PE router, you must configure an interface over which the VPN traffic travels between the PE and CE routers.

The sections that follow describe how to configure interfaces for VPNs:

General Configuration for VPN Routing

The configuration described in this section applies to all types of VPNs. For Layer 3 VPNs and carrier-of-carriers VPNs, complete the configuration described in this section before proceeding to the interface configuration sections specific to those topics.

To configure interfaces for VPN routing, include the `interface` statement:

```
interface interface-name;
```

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`*`]`

Specify both the physical and logical portions of the interface name, in the following format:

```
physical.logical
```

For example, in `at-1/2/1.2`, `at-1/2/1` is the physical portion of the interface name and `2` is the logical portion. If you do not specify the logical portion of the interface name, the value `0` is set by default.

A logical interface can be associated with only one routing instance. If you enable a routing protocol on all instances by specifying `interfaces all` when configuring the master instance of the protocol at the `[edit protocols]` hierarchy level, and if you configure a specific interface for VPN routing at the `[edit routing-instances` *`routing-instance-name`*`]` hierarchy level or at the `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`*`]` hierarchy level, the latter interface statement takes precedence and the interface is used exclusively for the VPN.

If you explicitly configure the same interface name at the `[edit protocols]` hierarchy level and at either the `[edit routing-instances` *`routing-instance-name`*`]` or `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`*`]` hierarchy levels, an attempt to commit the configuration fails.

Configuring Interfaces for Layer 3 VPNs

When you configure the Layer 3 VPN interfaces at the `[edit interfaces]` hierarchy level, you must also configure `family inet` when configuring the logical interface:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
    }
}
```

Configuring Interfaces for Carrier-of-Carriers VPNs

When you configure carrier-of-carriers VPNs, you need to configure the `family mpls` statement in addition to the `family inet` statement for the interfaces between the PE and CE routers. For carrier-of-carriers VPNs, configure the logical interface as follows:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
        family mpls;
    }
}
```

If you configure `family mpls` on the logical interface and then configure this interface for a non-carrier-of-carriers routing instance, the `family mpls` statement is automatically removed from the configuration for the logical interface, since it is not needed.

Configuring Unicast RPF on VPN Interfaces

For VPN interfaces that carry IP version 4 or version 6 (IPv4 or IPv6) traffic, you can reduce the impact of denial-of-service (DoS) attacks by configuring unicast reverse path forwarding (RPF). Unicast RPF helps determine the source of attacks and rejects packets from unexpected source addresses on interfaces where unicast RPF is enabled.

You can configure unicast RPF on a VPN interface by enabling unicast RPF on the interface and including the `interface` statement at the `[edit routing-instances routing-instance-name]` hierarchy level.

You cannot configure unicast RPF on the core-facing interfaces. You can only configure unicast RPF on the CE router-to-PE router interfaces on the PE router. However, for virtual-router routing instances, unicast RPF is supported on all interfaces you specify in the routing instance.

For information about how to configure unicast RPF on VPN interfaces, see Understanding Unicast RPF (Routers).

Configuring the Route Distinguisher

Each routing instance that you configure on a PE router must have a unique route distinguisher associated with it. VPN routing instances need a route distinguisher to help BGP to distinguish between potentially identical network layer reachability information (NLRI) messages received from different VPNs. If you configure different VPN routing instances with the same route distinguisher, the commit fails.

For Layer 2 VPNs and VPLS, if you have configured the `l2vpn-use-bgp-rules` statement, you must configure a unique route distinguisher for each PE router participating in a specific routing instance.

For other types of VPNs, we recommend that you use a unique route distinguisher for each PE router participating in the routing instance. Although you can use the same route distinguisher on all PE routers for the same VPN routing instance (except for Layer 2 VPNs and VPLS), if you use a unique route distinguisher, you can determine the CE router from which a route originated within the VPN.

To configure a route distinguisher on a PE router, include the `route-distinguisher` statement:

```
route-distinguisher (as-number:number | ip-address:number);
```

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

The route distinguisher is a 6-byte value that you can specify in one of the following formats:

- *as-number:number*, where *as-number* is an autonomous system (AS) number (a 2-byte value) and *number* is any 4-byte value. The AS number can be in the range 1 through 65,535. We recommend that you use an Internet Assigned Numbers Authority (IANA)-assigned, nonprivate AS number, preferably the Internet service provider's (ISP's) own or the customer's own AS number.

- *ip-address*:*number*, where *ip-address* is an IP address (a 4-byte value) and *number* is any 2-byte value. The IP address can be any globally unique unicast address. We recommend that you use the address that you configure in the `router-id` statement, which is a nonprivate address in your assigned prefix range.

Configuring Automatic Route Distinguishers

If you configure the `route-distinguisher-id` statement at the `[edit routing-options]` hierarchy level, a route distinguisher is automatically assigned to the routing instance. If you also configure the `route-distinguisher` statement in addition to the `route-distinguisher-id` statement, the value configured for `route-distinguisher` supersedes the value generated from `route-distinguisher-id`.

To assign a route distinguisher automatically, include the `route-distinguisher-id` statement:

```
route-distinguisher-id ip-address;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-options]`
- `[edit logical-systems logical-system-name routing-options]`

A type 1 route distinguisher is automatically assigned to the routing instance using the format *ip-address*:*number*. The IP address is specified by the `route-distinguisher-id` statement and the number is unique for the routing instance.

Configuring Virtual-Router Routing Instances in VPNs

IN THIS SECTION

- Configuring a Routing Protocol Between the Service Provider Routers | **16**
- Configuring Logical Interfaces Between Participating Routers | **17**

A virtual-router routing instance, like a VRF routing instance, maintains separate routing and forwarding tables for each instance. However, many of the configuration steps required for VRF routing instances are not required for virtual-router routing instances. Specifically, you do not need to configure a route distinguisher, a routing table policy (the `vrf-export`, `vrf-import`, and `route-distinguisher` statements), or MPLS between the service provider routers.

Configure a virtual-router routing instance by including the following statements:

```
description text;
instance-type virtual-router;
interface interface-name;
protocols { ... }
```

You can include these statements at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`*`]`

The following sections explain how to configure a virtual-router routing instance:

Configuring a Routing Protocol Between the Service Provider Routers

The service provider routers need to be able to exchange routing information. You can configure the following protocols for the virtual-router routing instance `protocols` statement configuration at the `[edit routing-instances` *`routing-instance-name`*`]` hierarchy level:

- BGP
- IS-IS
- LDP
- OSPF
- Protocol Independent Multicast (PIM)
- RIP

You can also configure static routes.

IBGP route reflection is not supported for virtual-router routing instances.

If you configure LDP under a virtual-router instance, LDP routes are placed by default in the routing instance's inet.0 and inet.3 routing tables (for example, sample.inet.0 and sample.inet.3). To restrict LDP routes to only the routing instance's inet.3 table, include the `no-forwarding` statement:

```
no-forwarding;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols ldp]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols ldp]`

When you restrict the LDP routes to only the inet.3 routing table, the corresponding IGP route in the inet.0 routing table can be redistributed and advertised into other routing protocols.

For information about routing tables, see Understanding Junos OS Routing Tables.

Configuring Logical Interfaces Between Participating Routers

You must configure an interface to each customer router participating in the routing instance and to each P router participating in the routing instance. Each virtual-router routing instance requires its own separate logical interfaces to all P routers participating in the instance. To configure interfaces for virtual-router instances, include the `interface` statement:

```
interface interface-name;
```

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`*`]`

Specify both the physical and logical portions of the interface name, in the following format:

```
physical.logical
```

For example, in `at-1/2/1.2`, `at-1/2/1` is the physical portion of the interface name and `2` is the logical portion. If you do not specify the logical portion of the interface name, `0` is set by default.

You must also configure the interfaces at the `[edit interfaces]` hierarchy level.

One method of providing this logical interface between the provider routers is by configuring tunnels between them. You can configure IP Security (IPsec), generic routing encapsulation (GRE), or IP-IP tunnels between the provider routers, terminating the tunnels at the virtual-router instance.

For information about how to configure tunnels and interfaces, see the Junos OS Services Interfaces Library for Routing Devices.

Configuring Path MTU Checks for VPN Routing Instances

IN THIS SECTION

- Enabling Path MTU Checks for a VPN Routing Instance | 18
- Assigning an IP Address to the VPN Routing Instance | 19

When you enable an MTU check, the routing platform sends an Internet Control Message Protocol (ICMP) message when a packet traversing the routing instance exceeds the MTU size and has the `do-not-fragment` bit set. The ICMP message uses the VRF local address as its source address.

For an MTU check to work in a routing instance, you must both include the `vrf-mtu-check` statement at the `[edit chassis]` hierarchy level and assign at least one interface containing an IP address to the routing instance.

For more information about the path MTU check, see the Junos OS Administration Library for Routing Devices.

To configure path MTU checks, do the tasks described in the following sections:

Enabling Path MTU Checks for a VPN Routing Instance

To enable path checks on the outgoing interface for unicast traffic routed on a VRF or virtual-router routing instance, include the `vrf-mtu-check` statement at the `[edit chassis]` hierarchy level:

```
[edit chassis]
vrf-mtu-check;
```

Assigning an IP Address to the VPN Routing Instance

To ensure that the path MTU check functions properly, at least one IP address must be associated with each VRF or virtual-router routing instance. If an IP address is not associated with the routing instance, ICMP reply messages cannot be sent.

Typically, the VRF or virtual-router routing instance IP address is drawn from among the IP addresses associated with interfaces configured for that routing instance. If none of the interfaces associated with a VRF or virtual-router routing instance is configured with an IP address, you need to explicitly configure a logical loopback interface with an IP address. This interface must then be associated with the routing instance. See *Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs* for details.

CHAPTER 3

Distributing Routes in VPNs

IN THIS CHAPTER

- Enabling Routing Information Exchange for VPNs | 20
- Configuring IBGP Sessions Between PE Routers in VPNs | 20
- Configuring Aggregate Labels for VPNs | 22
- Configuring a Signaling Protocol and LSPs for VPNs | 23
- Configuring Policies for the VRF Table on PE Routers in VPNs | 28
- Configuring the Route Origin for VPNs | 36

Enabling Routing Information Exchange for VPNs

For Layer 2 VPNs, Layer 3 VPNs, virtual-router routing instances, VPLS, EVPNs, and Layer 2 circuits to function properly, the service provider’s PE and P routers must be able to exchange routing information. For this to happen, you must configure either an IGP (such as OSPF or IS-IS) or static routes on these routers. You configure the IGP on the master instance of the routing protocol process at the `[edit protocols]` hierarchy level, not within the routing instance used for the VPN—that is, not at the `[edit routing-instances]` hierarchy level.

When you configure the PE router, do not configure any summarization of the PE router’s loopback addresses at the area boundary. Each PE router’s loopback address should appear as a separate route.

Configuring IBGP Sessions Between PE Routers in VPNs

You must configure an IBGP session between the PE routers to allow the PE routers to exchange information about routes originating and terminating in the VPN. The PE routers rely on this information to determine which labels to use for traffic destined for remote sites.

Configure an IBGP session for the VPN as follows:

```
[edit protocols]
bgp {
    group group-name {
        type internal;
        local-address ip-address;
        family evpn {
            signaling;
        }
        family (inet-vpn | inet6-vpn) {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        neighbor ip-address;
    }
}
```

The IP address in the `local-address` statement is the address of the loopback interface on the local PE router. The IBGP session for the VPN runs through the loopback address. (You must also configure the loopback interface at the `[edit interfaces]` hierarchy level.)

The IP address in the `neighbor` statement is the loopback address of the neighboring PE router. If you are using RSVP signaling, this IP address is the same address you specify in the `to` statement at the `[edit mpls label-switched-path` *`lsp-path-name`*`]` hierarchy level when you configure the MPLS LSP.

The `family` statement allows you to configure the IBGP session for Layer 2 VPNs, VPLS, EVPNs or for Layer 3 VPNs.

- To configure an IBGP session for Layer 2 VPNs and VPLS, include the `signaling` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level:

  ```
  [edit protocols bgp group group-name family l2vpn]
  signaling;
  ```

- To configure an IBGP session for EVPNs, include the `signaling` statement at the `[edit protocols bgp group group-name family evpn]` hierarchy level:

```
[edit protocols bgp group group-name family evpn]
signaling;
```

- To configure an IPv4 IBGP session for Layer 3 VPNs, configure the `unicast` statement at the `[edit protocols bgp group group-name family inet-vpn]` hierarchy level:

```
[edit protocols bgp group group-name family inet-vpn]
unicast;
```

- To configure an IPv6 IBGP session for Layer 3 VPNs, configure the `unicast` statement at the `[edit protocols bgp group group-name family inet6-vpn]` hierarchy level:

```
[edit protocols bgp group group-name family inet6-vpn]
unicast;
```

NOTE: You can configure both `family inet` and `family inet-vpn` or both `family inet6` and `family inet6-vpn` within the same peer group. This allows you to enable support for both IPv4 and IPv4 VPN routes or both IPv6 and IPv6 VPN routes within the same peer group.

Configuring Aggregate Labels for VPNs

Aggregate labels for VPNs allow a Juniper Networks routing platform to aggregate a set of incoming labels (labels received from a peer router) into a single forwarding label that is selected from the set of incoming labels. The single forwarding label corresponds to a single next hop for that set of labels. Label aggregation reduces the number of VPN labels that the router must examine.

For a set of labels to share an aggregate forwarding label, they must belong to the same forwarding equivalence class (FEC). The labeled packets must have the same destination egress interface.

Including the `community community-name` statement with the `aggregate-label` statement lets you specify prefixes with a common origin community. Set by policy on the peer PE, these prefixes represent an FEC on the peer PE router.

CAUTION: If the target community is set by mistake instead of the origin community, forwarding problems at the egress PE can result. All prefixes from the peer PE will appear to be in the same FEC, resulting in a single inner label for all CE routers behind a given PE in the same VPN.

To work with route reflectors in Layer 3 VPN networks, the router aggregates a set of incoming labels only when the routes:

- Are received from the same peer router
- Have the same site of origin community
- Have the same next hop

The next hop requirement is important because route reflectors forward routes originated from different BGP peers to another BGP peer without changing the next hop of those routes.

To configure aggregate labels for VPNs, include the *aggregate-label* statement:

```
aggregate-label {
    community community-name;
}
```

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

For information about how to configure a community, see Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions.

Configuring a Signaling Protocol and LSPs for VPNs

IN THIS SECTION

- Using LDP for VPN Signaling | 24
- Using RSVP for VPN Signaling | 26

For VPNs to function, you must enable a signaling protocol, either the LDP or RSVP on the provider edge (PE) routers and on the provider (P) routers. You also need to configure label-switched paths (LSPs) between the ingress and egress routers. In a typical VPN configuration, you need to configure LSPs from each PE router to all of the other PE routers participating in the VPN in a full mesh.

To enable a signaling protocol, perform the steps in one of the following sections:

Using LDP for VPN Signaling

To use LDP for VPN signaling, perform the following steps on the PE and provider (P) routers:

1. Configure LDP on the interfaces in the core of the service provider’s network by including the `ldp` statement at the `[edit protocols]` hierarchy level.

   You need to configure LDP only on the interfaces between PE routers or between PE and P routers. You can think of these as the “core-facing” interfaces. You do not need to configure LDP on the interface between the PE and customer edge (CE) routers.

   ```
   [edit]
   protocols {
       ldp {
           interface type-fpc/pic/port;
       }
   }
   ```

2. Configure the MPLS address family on the interfaces on which you enabled LDP (the interfaces you configured in Step "1" on page 24) by including the `family mpls` statement at the `[edit interfaces` *type-fpc/pic/port* `unit` *logical-unit-number*`]` hierarchy level.

   ```
   [edit]
   interfaces {
       type-fpc/pic/port {
           unit logical-unit-number {
               family mpls;
           }
       }
   }
   ```

3. Configure OSPF or IS-IS on each PE and P router.

   You configure these protocols at the master instance of the routing protocol, not within the routing instance used for the VPN.

- To configure OSPF, include the `ospf` statement at the `[edit protocols]` hierarchy level. At a minimum, you must configure a backbone area on at least one of the router's interfaces.

```
[edit]
protocols {
    ospf {
        area 0.0.0.0 {
            interface type-fpc/pic/port;
        }
    }
}
```

- To configure IS-IS, include the `isis` statement at the `[edit protocols]` hierarchy level and configure the loopback interface and International Organization for Standardization (ISO) family at the `[edit interfaces]` hierarchy level. At a minimum, you must enable IS-IS on the router, configure a network entity title (NET) on one of the router's interfaces (preferably the loopback interface, lo0), and configure the ISO family on all interfaces on which you want IS-IS to run. When you enable IS-IS, Level 1 and Level 2 are enabled by default. The following is the minimum IS-IS configuration. In the `address` statement, *address* is the NET.

```
[edit]
interfaces {
    lo0 {
        unit logical-unit-number {
            family iso {
                address address;
            }
        }
    }
    type-fpc/pic/port {
        unit logical-unit-number {
            family iso;
        }
    }
}
protocols {
    isis {
        interface all;
    }
}
```

Using RSVP for VPN Signaling

To use RSVP for VPN signaling, perform the following steps:

1. On each PE router, configure traffic engineering.

   To do this, you must configure an interior gateway protocol (IGP) that supports traffic engineering (either IS-IS or OSPF) and enable traffic engineering support for that protocol.

   To enable OSPF traffic engineering support, include the `traffic-engineering` statement at the `[edit protocols ospf]` hierarchy level:

   ```
   [edit protocols ospf]
   traffic-engineering {
       shortcuts;
   }
   ```

   For IS-IS, traffic engineering support is enabled by default.

2. On each PE and P router, enable RSVP on the interfaces that participate in the label-switched path (LSP).

   On the PE router, these interfaces are the ingress and egress points to the LSP. On the P router, these interfaces connect the LSP between the PE routers. Do not enable RSVP on the interface between the PE and the CE routers, because this interface is not part of the LSP.

   To configure RSVP on the PE and P routers, include the `interface` statement at the `[edit protocols rsvp]` hierarchy level. Include one `interface` statement for each interface on which you are enabling RSVP.

   ```
   [edit protocols]
   rsvp {
       interface interface-name;
       interface interface-name;
   }
   ```

3. On each PE router, configure an MPLS LSP to the PE router that is the LSP's egress point.

   To do this, include the `interface` and `label-switched-path` statements at the `[edit protocols mpls]` hierarchy level:

   ```
   [edit protocols]
   mpls {
       interface interface-name;
   ```

```
    label-switched-path path-name {
        to ip-address;
    }
}
```

In the `to` statement, specify the address of the LSP's egress point, which is an address on the remote PE router.

In the `interface` statement, specify the name of the interface (both the physical and logical portions). Include one `interface` statement for the interface associated with the LSP.

When you configure the logical portion of the same interface at the `[edit interfaces]` hierarchy level, you must also configure the `family inet` and `family mpls` statements:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
        family mpls;
    }
}
```

4. On all P routers that participate in the LSP, enable MPLS by including the `interface` statement at the `[edit mpls]` hierarchy level.

   Include one `interface` statement for each connection to the LSP.

```
[edit]
mpls {
    interface interface-name;
    interface interface-name;
}
```

5. Enable MPLS on the interface between the PE and CE routers by including the `interface` statement at the `[edit mpls]` hierarchy level.

   Doing this allows the PE router to assign an MPLS label to traffic entering the LSP or to remove the label from traffic exiting the LSP.

```
[edit]
mpls {
    interface interface-name;
}
```

For information about configuring MPLS, see the Configuring the Ingress Router for MPLS-Signaled LSPs.

RELATED DOCUMENTATION

Configuring the Ingress Router for MPLS-Signaled LSPs

Configuring Policies for the VRF Table on PE Routers in VPNs

IN THIS SECTION

- Configuring the Route Target | 28
- Configuring the Route Origin | 29
- Configuring an Import Policy for the PE Router's VRF Table | 30
- Configuring an Export Policy for the PE Router's VRF Table | 32
- Applying Both the VRF Export and the BGP Export Policies | 34
- Configuring a VRF Target | 34

On each PE router, you must define policies that define how routes are imported into and exported from the router's VRF table. In these policies, you must define the route target, and you can optionally define the route origin.

To configure policy for the VRF tables, you perform the steps in the following sections:

Configuring the Route Target

As part of the policy configuration for the VPN routing table, you must define a route target, which defines which VPN the route is a part of. When you configure different types of VPN services (Layer 2 VPNs, Layer 3 VPNs, EVPNs, or VPLS) on the same PE router, be sure to assign unique route target values to avoid the possibility of adding route and signaling information to the wrong VPN routing table.

To configure the route target, include the `target` option in the `community` statement:

```
community name members target:community-id;
```

You can include this statement at the following hierarchy levels:

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

name is the name of the community.

community-id is the identifier of the community. Specify it in one of the following formats:

- *as-number*:*number*, where *as-number* is an AS number (a 2-byte value) and *number* is a 4-byte community value. The AS number can be in the range 1 through 65,535. We recommend that you use an IANA-assigned, nonprivate AS number, preferably the ISP's own or the customer's own AS number. The community value can be a number in the range 0 through 4,294,967,295 (2^{32} – 1).
- *ip-address*:*number*, where *ip-address* is an IPv4 address (a 4-byte value) and *number* is a 2-byte community value. The IP address can be any globally unique unicast address. We recommend that you use the address that you configure in the `router-id` statement, which is a nonprivate address in your assigned prefix range. The community value can be a number in the range 1 through 65,535.

Configuring the Route Origin

In the import and export policies for the PE router's VRF table, you can optionally assign the route origin (also known as the site of origin) for a PE router's VRF routes using a VRF export policy applied to multiprotocol external BGP (MP-EBGP) VPN IPv4 route updates sent to other PE routers.

Matching on the assigned route origin attribute in a receiving PE's VRF import policy helps ensure that VPN-IPv4 routes learned through MP-EBGP updates from one PE are not reimported to the same VPN site from a different PE connected to the same site.

To configure a route origin, complete the following steps:

1. Include the `community` statement with the `origin` option:

   ```
   community name members origin:community-id;
   ```

   You can include this statement at the following hierarchy levels:

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

name is the name of the community.

community-id is the identifier of the community. Specify it in one of the following formats:

- *as-number*:*number*, where *as-number* is an AS number (a 2-byte value) and *number* is a 4-byte community value. The AS number can be in the range 1 through 65,535. We recommend that you use an IANA-assigned, nonprivate AS number, preferably the ISP's own or the customer's own AS number. The community value can be a number in the range 0 through 4,294,967,295 (2^{32} – 1).

- *ip-address*:*number*, where *ip-address* is an IPv4 address (a 4-byte value) and *number* is a 2-byte community value. The IP address can be any globally unique unicast address. We recommend that you use the address that you configure in the `router-id` statement, which is a nonprivate address in your assigned prefix range. The community value can be a number in the range 1 through 65,535.

2. Include the community in the import policy for the PE router's VRF table by configuring the `community` statement with the *community-id* identifier defined in Step "1" on page 29 at the `[edit policy-options policy-statement` *import-policy-name* `term` *import-term-name* `from]` hierarchy level. See "Configuring an Import Policy for the PE Router's VRF Table" on page 30.

   If the policy's `from` clause does not specify a community condition, the `vrf-import` statement in which the policy is applied cannot be committed. The Junos OS commit operation does not pass the validation check.

3. Include the community in the export policy for the PE router's VRF table by configuring the `community` statement with the *community-id* identifier defined in Step "1" on page 29 at the `[edit policy-options policy-statement` *export-policy-name* `term` *export-term-name* `then]` hierarchy level. See "Configuring an Export Policy for the PE Router's VRF Table" on page 32.

See *Configuring the Route Origin for VPNs* for a configuration example.

Configuring an Import Policy for the PE Router's VRF Table

Each VPN can have a policy that defines how routes are imported into the PE router's VRF table. An import policy is applied to routes received from other PE routers in the VPN. A policy must evaluate all routes received over the IBGP session with the peer PE router. If the routes match the conditions, the route is installed in the PE router's *routing-instance-name*`.inet.0` VRF table. An import policy must contain a second term that rejects all other routes.

Unless an import policy contains only a `then reject` statement, it must include a reference to a community. Otherwise, when you try to commit the configuration, the commit fails. You can configure multiple import policies.

An import policy determines what to import to a specified VRF table based on the VPN routes learned from the remote PE routers through IBGP. The IBGP session is configured at the `[edit protocols bgp]` hierarchy level. If you also configure an import policy at the `[edit protocols bgp]` hierarchy level, the import policies at the `[edit policy-options]` hierarchy level and the `[edit protocols bgp]` hierarchy level are combined through a logical AND operation. This allows you to filter traffic as a group.

To configure an import policy for the PE router's VRF table, follow these steps:

1. To define an import policy, include the `policy-statement` statement. For all PE routers, an import policy must always include the `policy-statement` statement, at a minimum:

   ```
   policy-statement import-policy-name {
       term import-term-name {
           from {
               protocol bgp;
               community community-id;
           }
           then accept;
       }
       term term-name {
           then reject;
       }
   }
   ```

   You can include the `policy-statement` statement at the following hierarchy levels:

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

   The *`import-policy-name`* policy evaluates all routes received over the IBGP session with the other PE router. If the routes match the conditions in the `from` statement, the route is installed in the PE router's *routing-instance-name*.inet.0 VRF table. The second term in the policy rejects all other routes.

   For more information about creating policies, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

2. You can optionally use a regular expression to define a set of communities to be used for the VRF import policy.

   For example you could configure the following using the `community` statement at the `[edit policy-options policy-statement policy-statement-name]` hierarchy level:

   ```
   [edit policy-options vrf-import-policy-sample]
   community high-priority members *:50
   ```

Note that you cannot configure a regular expression as a part of a route target extended community. For more information about how to configure regular expressions for communities, see How BGP Communities and Extended Communities Are Evaluated in Routing Policy Match Conditions.

3. To configure an import policy, include the `vrf-import` statement:

```
vrf-import import-policy-name;
```

   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]`

Configuring an Export Policy for the PE Router's VRF Table

Each VPN can have a policy that defines how routes are exported from the PE router's VRF table. An export policy is applied to routes sent to other PE routers in the VPN. An export policy must evaluate all routes received over the routing protocol session with the CE router. (This session can use the BGP, OSPF, or Routing Information Protocol [RIP] routing protocols, or static routes.) If the routes match the conditions, the specified community target (which is the route target) is added to them and they are exported to the remote PE routers. An export policy must contain a second term that rejects all other routes.

Export policies defined within the VPN routing instance are the only export policies that apply to the VRF table. Any export policy that you define on the IBGP session between the PE routers has no effect on the VRF table. You can configure multiple export policies.

To configure an export policy for the PE router's VRF table, follow these steps:

1. For all PE routers, an export policy must distribute VPN routes to and from the connected CE routers in accordance with the type of routing protocol that you configure between the CE and PE routers within the routing instance.

   To define an export policy, include the `policy-statement` statement. An export policy must always include the `policy-statement` statement, at a minimum:

```
policy-statement export-policy-name {
    term export-term-name {
        from protocol (bgp | ospf | rip | static);
        then {
            community add community-id;
            accept;
        }
    }
    term term-name {
        then reject;
    }
}
```

NOTE: Configuring the `community add` statement is a requirement for Layer 2 VPN VRF export policies. If you change the `community add` statement to the `community set` statement, the router at the egress of the Layer 2 VPN link might drop the connection.

NOTE: When configuring draft-rosen multicast VPNs operating in source-specific mode and using the `vrf-export` statement to specify the export policy, the policy must have a term that accepts routes from the vrf-name.mdt.0 routing table. This term ensures proper PE autodiscovery using the `inet-mdt` address family.

When configuring draft-rosen multicast VPNs operating in source-specific mode and using the `vrf-target` statement, the VRF export policy is automatically generated and automatically accepts routes from the vrf-name.mdt.0 routing table.

You can include the `policy-statement` statement at the following hierarchy levels:

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

The *`export-policy-name`* policy evaluates all routes received over the routing protocol session with the CE router. (This session can use the BGP, OSPF, or RIP routing protocols, or static routes.) If the routes match the conditions in the `from` statement, the community target specified in the `then community add` statement is added to them and they are exported to the remote PE routers. The second term in the policy rejects all other routes.

For more information about creating policies, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

2. To apply the policy, include the `vrf-export` statement:

```
vrf-export export-policy-name;
```

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*]`

Applying Both the VRF Export and the BGP Export Policies

When you apply a VRF export policy as described in "Configuring an Export Policy for the PE Router's VRF Table" on page 32, routes from VPN routing instances are advertised to other PE routers based on this policy, whereas the BGP export policy is ignored.

If you include the `vpn-apply-export` statement in the BGP configuration, both the VRF export and BGP group or neighbor export policies are applied (VRF first, then BGP) before routes are advertised in the VPN routing tables to other PE routers.

NOTE: When a PE device is also acting as a Route Reflector (RR) or an Autonomous system boundary router (ASBR) in a Carrier-over-Carrier or inter-AS VPN, the next-hop manipulation in the vrf-export policy is ignored.

When you include the `vpn-apply-export` statement, be aware of the following:

- Routes imported into the bgp.l3vpn.0 routing table retain the attributes of the original routes (for example, an OSPF route remains an OSPF route even when it is stored in the bgp.l3vpn.0 routing table). You should be aware of this when you configure an export policy for connections between an IBGP PE router and a PE router, a route reflector and a PE router, or AS boundary router (ASBR) peer routers.

- By default, all routes in the bgp.l3vpn.0 routing table are exported to the IBGP peers. If the last statement of the export policy is deny all and if the export policy does not specifically match on routes in the bgp.l3vpn.0 routing table, no routes are exported.

To apply both the VRF export and BGP export policies to VPN routes, include the `vpn-apply-export` statement:

```
vpn-apply-export;
```

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

Configuring a VRF Target

Including the `vrf-target` statement in the configuration for a VRF target community causes default VRF import and export policies to be generated that accept and tag routes with the specified target community. You can still create more complex policies by explicitly configuring VRF import and export policies. These policies override the default policies generated when you configure the `vrf-target` statement.

If you do not configure the `import` and `export` options of the `vrf-target` statement, the specified community string is applied in both directions. The `import` and `export` keywords give you more flexibility, allowing you to specify a different community for each direction.

The syntax for the VRF target community is not a name. You must specify it in the format `target:`*`x`*`:`*`y`*. A community name cannot be specified because this would also require you to configure the community members for that community using the `policy-options` statement. If you define the `policy-options` statements, then you can just configure VRF import and export policies as usual. The purpose of the `vrf-target` statement is to simplify the configuration by allowing you to configure most statements at the `[edit routing-instances]` hierarchy level.

To configure a VRF target, include the `vrf-target` statement:

```
vrf-target community;
```

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`*`]`

An example of how you might configure the `vrf-target` statement follows:

```
[edit routing-instances sample]
vrf-target target:69:102;
```

To configure the `vrf-target` statement with the `export` and `import` options, include the following statements:

```
vrf-target {
    export community-name;
    import community-name;
}
```

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`*`]`

Configuring the Route Origin for VPNs

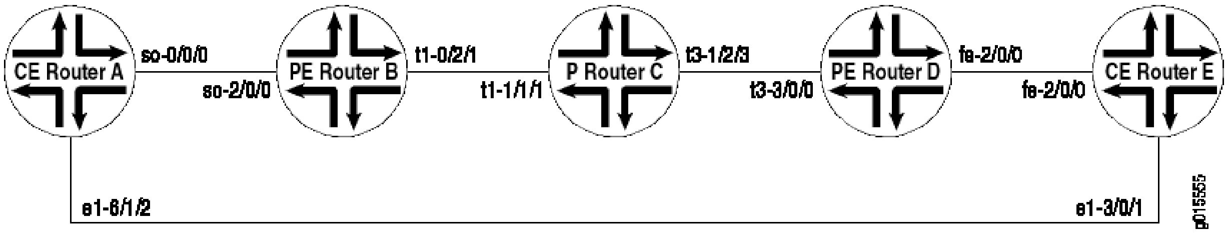
IN THIS SECTION

- Configuring the Site of Origin Community on CE Router A | 36
- Configuring the Community on CE Router A | 37
- Applying the Policy Statement on CE Router A | 38
- Configuring the Policy on PE Router D | 39
- Configuring the Community on PE Router D | 39
- Applying the Policy on PE Router D | 39

You can use route origin to prevent routes learned from one customer edge (CE) router marked with origin community from being advertised back to it from another CE router in the same AS.

In the example, the route origin is used to prevent routes learned from CE Router A that are marked with origin community from being advertised back to CE Router E by AS 200. The example topology is shown in Figure 3 on page 36.

Figure 3: Network Topology of Site of Origin Example



In this topology, CE Router A and CE Router E are in the same AS (AS200). They use EBGP to exchange routes with their respective provider edge (PE) routers, PE Router B and PE Router D. The two CE routers have a back connection.

The following sections describe how to configure the route origin for a group of VPNs:

Configuring the Site of Origin Community on CE Router A

The following section describes how to configure CE Router A to advertise routes with a site of origin community to PE Router B for this example.

NOTE: In this example, direct routes are configured to be advertised, but any route can be configured.

Configure a policy to advertise routes with `my-soo` community on CE Router A as follows:

```
[edit]
policy-options {
    policy-statement export-to-my-isp {
        term a {
            from {
                protocol direct;
            }
            then {
                community add my-soo;
                accept;
            }
        }
    }
}
```

Configuring the Community on CE Router A

Configure the `my-soo` community on CE Router A as follows:

```
[edit]
policy-options {
    community my-soo {
        members origin:100:1;
    }
}
```

Applying the Policy Statement on CE Router A

Apply the export-to-my-isp policy statement as an export policy to the EBGP peering on the CE Router A as follows:

```
[edit]
protocols {
    bgp {
        group my_isp {
            export export-to-my-isp;
        }
    }
}
```

When you issue the `show route receive-protocol bgp detail` command, you should see the following routes originated from PE Router B with `my-soo` community:

```
user@host> show route receive-protocol bgp 10.12.99.2 detail
inet.0: 16 destinations, 16 routes (15 active, 0 holddown, 1 hidden)
inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
vpn_blue.inet.0: 8 destinations, 10 routes (8 active, 0 holddown, 0 hidden)
* 10.12.33.0/30 (2 entries, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
  10.12.99.0/30 (2 entries, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
* 10.255.71.177/32 (1 entry, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
* 192.168.64.0/21 (1 entry, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
mpls.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
```

```
__juniper_private1__.inet6.0: 1 destinations, 1 routes (1 active, 0 holddown, 0
hidden)
```

Configuring the Policy on PE Router D

Configure a policy on PE Router D that prevents routes with `my-soo` community tagged by CE Router A from being advertised to CE Router E as follows:

```
[edit]
policy-options {
    policy-statement soo-ce1-policy {
        term a {
            from {
                community my-soo;
                then {
                    reject;
                }
            }
        }
    }
}
```

Configuring the Community on PE Router D

Configure the community on PE Router D as follows:

```
[edit]
policy-options {
    community my-soo {
        members origin:100:1;
    }
}
```

Applying the Policy on PE Router D

To prevent routes learned from CE Router A from being advertised to CE Router E (the two routers can communicate these routes directly), apply the `soo-ce1-policy` policy statement as an export policy to the PE Router D and CE Router E EBGP session `vpn_blue`.

View the EBGP session on PE Router D using the `show routing-instances` command.

```
user@host# show routing-instances
vpn_blue {
    instance-type vrf;
    interface fe-2/0/0.0;
    vrf-target target:100:200;
    protocols {
        bgp {
            group ce2 {
                advertise-peer-as;
                peer-as 100;
                neighbor 10.12.99.6;
            }
        }
    }
}
```

Apply the `soo-ce1-policy` policy statement as an export policy to the PE Router D and CE Router E EBGP session `vpn_blue` as follows:

```
[edit routing-instances]
vpn_blue {
    protocols {
        bgp {
            group ce2{
                export soo-ce1-policy;
            }
        }
    }
}
```

CHAPTER 4

Distributing VPN Routes with Target Filtering

IN THIS CHAPTER

- Configuring BGP Route Target Filtering for VPNs | **41**
- Example: BGP Route Target Filtering for VPNs | **43**
- Example: Configuring BGP Route Target Filtering for VPNs | **46**
- Configuring Static Route Target Filtering for VPNs | **58**
- Understanding Proxy BGP Route Target Filtering for VPNs | **59**
- Example: Configuring Proxy BGP Route Target Filtering for VPNs | **60**
- Example: Configuring an Export Policy for BGP Route Target Filtering for VPNs | **82**
- Reducing Network Resource Use with Static Route Target Filtering for VPNs | **106**

Configuring BGP Route Target Filtering for VPNs

IN THIS SECTION

- BGP Route Target Filtering Overview | **42**
- Configuring BGP Route Target Filtering for VPNs | **42**

BGP route target filtering allows you to distribute VPN routes to only the routers that need them. In VPN networks without BGP route target filtering configured, BGP distributes all VPN routes to all VPN peer routers.

For more information about BGP route target filtering, see RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs).*

The following sections provide an overview of BGP route target filtering and how to configure it for VPNs:

BGP Route Target Filtering Overview

PE routers, unless they are configured as route reflectors or are running an EBGP session, discard any VPN routes that do not include a route target extended community as specified in the local VRF import policies. This is the default behavior of the Junos OS.

However, unless it is explicitly configured not to store VPN routes, any router configured either as a route reflector or border router for a VPN address family must store all of the VPN routes that exist in the service provider's network. Also, though PE routers can automatically discard routes that do not include a route target extended community, route updates continue to be generated and received.

By reducing the number of routers receiving VPN routes and route updates, BGP route target filtering helps to limit the amount of overhead associated with running a VPN. BGP route target filtering is most effective at reducing VPN-related administrative traffic in networks where there are many route reflectors or AS border routers that do not participate in the VPNs directly (not acting as PE routers for the CE devices).

BGP route target filtering uses standard UPDATE messages to distributes route target extended communities between routers. The use of UPDATE messages allows BGP to use its standard loop detection mechanisms, path selection, policy support, and database exchange implementation.

Configuring BGP Route Target Filtering for VPNs

BGP route target filtering is enabled through the exchange of the `route-target` address family, stored in the bgp.rtarget.0 routing table. Based on the `route-target` address family, the route target NLRI (address family indicator [AFI]=1, subsequent AFI [SAFI]=132) is negotiated with its peers.

On a system that has locally configured VRF instances, BGP automatically generates local routes corresponding to targets referenced in the `vrf-import` policies.

To configure BGP route target filtering, include the `family route-target` statement:

```
family route-target {
    advertise-default;
    external-paths number;
    prefix-limit number;
}
```

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

The `advertise-default`, `external-paths`, and `prefix-limit` statements affect the BGP route target filtering configuration as follows:

- The `advertise-default` statement causes the router to advertise the default route target route (0:0:0/0) and suppress all routes that are more specific. This can be used by a route reflector on BGP groups consisting of neighbors that act as PE routers only. PE routers often need to advertise all routes to the route reflector.

  Suppressing all route target advertisements other than the default route reduces the amount of information exchanged between the route reflector and the PE routers. The Junos OS further helps to reduce route target advertisement overhead by not maintaining dependency information unless a nondefault route is received.

- The `external-paths` statement (which has a default value of 1) causes the router to advertise the VPN routes that reference a given route target. The number you specify determines the number of external peer routers (currently advertising that route target) that receive the VPN routes.

- The `prefix-limit` statement limits the number of prefixes that can be received from a peer router.

The `route-target`, `advertise-default`, and `external-path` statements affect the RIB-OUT state and must be consistent between peer routers that share the same BGP group. The `prefix-limit` statement affects the receive side only and can have different settings between different peer routers in a BGP group.

RELATED DOCUMENTATION

Configuring the Route Origin for VPNs

Example: BGP Route Target Filtering for VPNs

BGP route target filtering is enabled by configuring the `family route-target` statement at the appropriate BGP hierarchy level. This statement enables the exchange of a new `route-target` address family, which is stored in the bgp.rtarget.0 routing table.

The following configuration illustrates how you could configure BGP route target filtering for a BGP group titled `to_vpn04`:

```
[edit]
protocols {
    bgp {
        group to_vpn04 {
            type internal;
            local-address 10.255.14.182;
            peer-as 200;
            neighbor 10.255.14.174 {
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
        }
    }
}
```

The following configuration illustrates how you could configure a couple of local VPN routing and forwarding (VRF) routing instances to take advantage of the functionality provided by BGP route target filtering. Based on this configuration, BGP would automatically generate local routes corresponding to the route targets referenced in the VRF import policies (note the targets defined by the `vrf-target` statements).

```
[edit]
routing-instances {
    vpn1 {
        instance-type vrf;
        interface t1-0/1/2.0;
        vrf-target target:200:101;
        protocols {
            ospf {
                export bgp-routes;
                area 0.0.0.0 {
                    interface t1-0/1/2.0;
                }
            }
        }
    }
    vpn2 {
        instance-type vrf;
        interface t1-0/1/2.1;
        vrf-target target:200:102;
        protocols {
            ospf {
                export bgp-routes;
                area 0.0.0.0 {
                    interface t1-0/1/2.1;
                }
            }
        }
    }
}
```

Issue the **show route table bgp.rtarget.0** show command to verify the BGP route target filtering configuration:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both
200:200:101/96
                    *[RTarget/5] 00:10:00
                       Local
200:200:102/96
                    *[RTarget/5] 00:10:00
                       Local
200:200:103/96
                    *[BGP/170] 00:09:48, localpref 100, from 10.255.14.174
                       AS path: I
                     > t3-0/0/0.0
200:200:104/96
                    *[BGP/170] 00:09:48, localpref 100, from 10.255.14.174
                       AS path: I
                     > t3-0/0/0.0
```

The `show` command display format for route target prefixes is:

```
AS number:route target extended community/length
```

The first number represents the autonomous system (AS) of the router that sent this advertisement. The remainder of the display follows the Junos `show` command convention for extended communities.

The output from the `show route table bgp-rtarget.0` command displays the locally generated and remotely generated routes.

The first two entries correspond to the route targets configured for the two local VRF routing instances (`vpn1` and `vpn2`):

- `200:200:101/96`—Community `200:101` in the `vpn1` routing instance
- `200:200:102/96`—Community `200:102` in the `vpn2` routing instance

The last two entries are prefixes received from a BGP peer:

- `200:200:103/96`—Tells the local router that routes tagged with this community (`200:103`) should be advertised to peer `10.255.14.174` through `t3-0/0/0.0`
- `200:200:104/96`—Tells the local router that routes tagged with this community (`200:104`) should be advertised to peer `10.255.14.174` through `t3-0/0/0.0`

Example: Configuring BGP Route Target Filtering for VPNs

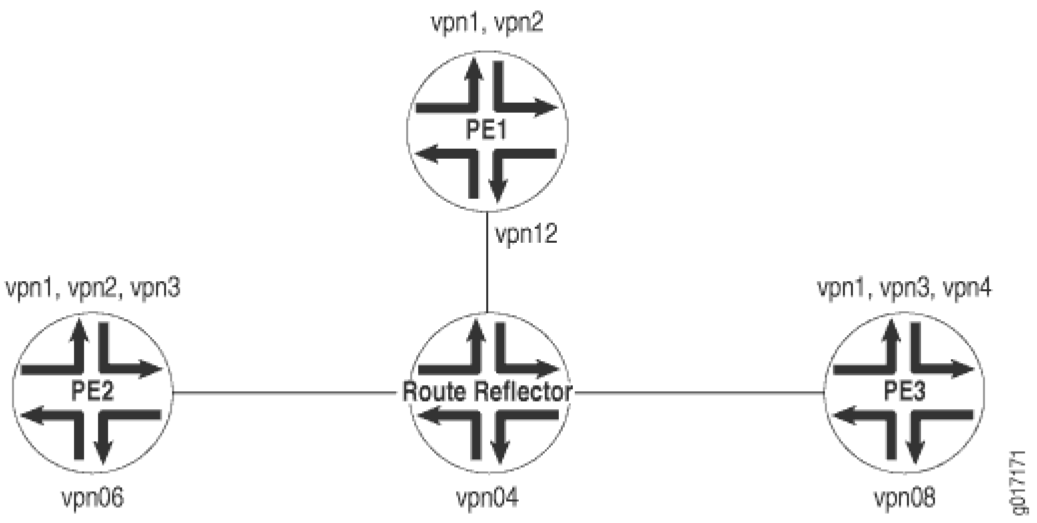
IN THIS SECTION

- Configure BGP Route Target Filtering on Router PE1 | **47**
- Configure BGP Route Target Filtering on Router PE2 | **49**
- Configure BGP Route Target Filtering on the Route Reflector | **53**
- Configure BGP Route Target Filtering on Router PE3 | **55**

BGP route target filtering reduces the number of routers that receive VPN routes and route updates, helping to limit the amount of overhead associated with running a VPN. BGP route target filtering is most effective at reducing VPN-related administrative traffic in networks where there are many route reflectors or AS border routers that do not participate in the VPNs directly (do not act as PE routers for the CE devices).

Figure 4 on page 47 illustrates the topology for a network configured with BGP route target filtering for a group of VPNs.

Figure 4: BGP Route Target Filtering Enabled for a Group of VPNs



The following sections describe how to configure BGP route target filtering for a group of VPNs:

Configure BGP Route Target Filtering on Router PE1

This section describes how to enable BGP route target filtering on Router PE1 for this example.

Configure the routing options on router PE1 as follows:

```
[edit]
routing-options {
    route-distinguisher-id 10.255.14.182;
    autonomous-system 198;
}
```

Configure the BGP protocol on Router PE1 as follows:

```
[edit]
protocols {
    bgp {
        group to_VPN_D {
            type internal;
            local-address 10.255.14.182;
            peer-as 198;
            neighbor 10.255.14.174 {
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
        }
    }
}
```

Configure the vpn1 routing instance as follows:

```
[edit]
routing-instances {
    vpn1 {
        instance-type vrf;
        interface t1-0/1/2.0;
        vrf-target target:198:101;
        protocols {
            ospf {
                export bgp-routes;
                area 0.0.0.0 {
                    interface t1-0/1/2.0;
                }
            }
        }
    }
}
```

Configure the vpn2 routing instance on Router PE1 as follows:

```
[edit]
routing-instances {
    vpn2 {
        instance-type vrf;
        interface t1-0/1/2.1;
        vrf-target target:198:102;
        protocols {
            ospf {
                export bgp-routes;
                area 0.0.0.0 {
                    interface t1-0/1/2.1;
                }
            }
        }
    }
}
```

Once you have implemented this configuration, you should see the following when you issue a `show route table bgp.rtarget.0` command:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 198.198:101/96
                   *[RTarget/5] 00:27:42
                      Local
                    [BGP/170] 00:27:30, localpref 100, from
 10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
 198.198:102/96
                   *[RTarget/5] 00:27:42
                      Local
                    [BGP/170] 00:27:30, localpref 100, from
 10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
 198.198:103/96
                   *[BGP/170] 00:27:30, localpref 100, from
 10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
 198.198:104/96
                   *[BGP/170] 00:27:30, localpref 100, from
 10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
```

Configure BGP Route Target Filtering on Router PE2

This section describes how to enable BGP route target filtering on Router PE2 for this example.

Configure the routing options on Router PE2 as follows:

```
[edit]
routing-options {
    route-distinguisher-id 10.255.14.176;
    autonomous-system 198;
}
```

Configure the BGP protocol on Router PE2 as follows:

```
[edit]
protocols {
    bgp {
        group to_vpn04 {
            type internal;
            local-address 10.255.14.176;
            peer-as 198;
            neighbor 10.255.14.174 {
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
        }
    }
}
```

Configure the `vpn1` routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
    vpn1 {
        instance-type vrf;
        interface t3-0/0/0.0;
        vrf-target target:198:101;
        protocols {
            bgp {
                group vpn1 {
                    type external;
                    peer-as 101;
                    as-override;
                    neighbor 10.49.11.2;
                }
            }
        }
    }
}
```

Configure the `vpn2` routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
    vpn2 {
        instance-type vrf;
        interface t3-0/0/0.1;
        vrf-target target:198:102;
        protocols {
            bgp {
                group vpn2 {
                    type external;
                    peer-as 102;
                    as-override;
                    neighbor 10.49.21.2;
                }
            }
        }
    }
}
```

Configure the `vpn3` routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
    vpn3 {
        instance-type vrf;
        interface t3-0/0/0.2;
        vrf-import vpn3-import;
        vrf-export vpn3-export;
        protocols {
            bgp {
                group vpn3 {
                    type external;
                    peer-as 103;
                    as-override;
                    neighbor 10.49.31.2;
                }
            }
        }
    }
}
```

Once you have configured router PE2 in this manner, you should see the following when you issue the `show route table bgp.rtarget.0` command:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 198.198:101/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 198.198:102/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 198.198:103/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 198.198:104/96
                    *[BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
```

```
                        AS path: I
                      > via t1-0/1/0.0
```

Configure BGP Route Target Filtering on the Route Reflector

This section illustrates how to enable BGP route target filtering on the route reflector for this example.

Configure the routing options on the route reflector as follows:

```
[edit]
routing-options {
    route-distinguisher-id 10.255.14.174;
    autonomous-system 198;
}
```

Configure the BGP protocol on the route reflector as follows:

```
[edit]
protocols {
    bgp {
        group rr-group {
            type internal;
            local-address 10.255.14.174;
            cluster 10.255.14.174;
            peer-as 198;
            neighbor 10.255.14.182 {
                description to_PE1_vpn12;
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
            neighbor 10.255.14.176 {
                description to_PE2_vpn06;
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
            neighbor 10.255.14.178 {
                description to_PE3_vpn08;
```

```
                  family inet-vpn {
                      unicast;
                  }
                  family route-target;
              }
          }
      }
  }
```

Once you have configured the route reflector in this manner, you should see the following when you issue the `show route table bgp.rtarget.0` command:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 8 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 198.198:101/96
                    *[BGP/170] 00:29:03, localpref 100, from
 10.255.14.176
                       AS path: I
                     > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
 10.255.14.178
                       AS path: I
                     > via t3-0/1/1.0
                     [BGP/170] 00:29:03, localpref 100, from
 10.255.14.182
                       AS path: I
                     > via t3-0/1/3.0
 198.198:102/96
                    *[BGP/170] 00:29:03, localpref 100, from
 10.255.14.176
                       AS path: I
                     > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
 10.255.14.182
                       AS path: I
                     > via t3-0/1/3.0
 198.198:103/96
                    *[BGP/170] 00:29:03, localpref 100, from
 10.255.14.176
                       AS path: I
                     > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
 10.255.14.178
                       AS path: I
                     > via t3-0/1/1.0
 198.198:104/96
                    *[BGP/170] 00:29:03, localpref 100, from
 10.255.14.178
                       AS path: I
                     > via t3-0/1/1.0
```

Configure BGP Route Target Filtering on Router PE3

The following section describes how to enable BGP route target filtering on Router PE3 for this example.

Configure the routing options on Router PE3 as follows:

```
[edit]
routing-options {
    route-distinguisher-id 10.255.14.178;
    autonomous-system 198;
}
```

Configure the BGP protocol on Router PE3 as follows:

```
[edit]
protocols {
    bgp {
        group to_vpn04 {
            type internal;
            local-address 10.255.14.178;
            peer-as 198;
            neighbor 10.255.14.174 {
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
        }
    }
}
```

Configure the `vpn1` routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
    vpn1 {
        instance-type vrf;
        interface t3-0/0/0.0;
        vrf-target target:198:101;
        protocols {
            rip {
                group vpn1 {
                    export bgp-routes;
                    neighbor t3-0/0/0.0;
                }
            }
        }
    }
}
```

Configure the `vpn3` routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
    vpn3 {
        instance-type vrf;
        interface t3-0/0/0.1;
        vrf-target target:198:103;
        protocols {
            rip {
                group vpn3 {
                    export bgp-routes;
                    neighbor t3-0/0/0.1;
                }
            }
        }
    }
}
```

Configure the `vpn4` routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
    vpn4 {
        instance-type vrf;
        interface t3-0/0/0.2;
        vrf-target target:198:104;
        protocols {
            rip {
                group vpn4 {
                    export bgp-routes;
                    neighbor t3-0/0/0.2;
                }
            }
        }
    }
}
```

Once you have configured Router PE3 in this manner, you should see the following when you issue the `show route table bgp.rtarget.0` command:

```
 user@host> show route table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 198.198:101/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 198.198:102/96
                    *[BGP/170] 00:29:29, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 198.198:103/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 198.198:104/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
```

Configuring Static Route Target Filtering for VPNs

The BGP VPN route target extended community (RFC 4360, *BGP Extended Communities Attribute*) is used to determine VPN membership. Static route target filtering helps to prevent resources from being consumed in portions of the network where the VPN routes are not needed due to the lack of member PE routers (RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*). Routers can originate routes into the RT-Constrain protocol to indicate their interest in receiving VPN routes containing route targets that match the RT-Constrain NLRI.

To configure static route target filtering for VPNs:

- Configure the `route-target-filter` statement at the `[edit routing-options rib bgp.rtarget.0 static]` hierarchy level.

  The following example illustrates how you could configure the *route-target-filter* statement:

  ```
  [edit routing-options rib bgp.rtarget.0 static]
  route-target-filter destination {
      group bgp-group;
      local;
      neighbor bgp-peer;
  }
  ```

- You can display route target filtering information using the `show bgp group rtf detail` command.

Understanding Proxy BGP Route Target Filtering for VPNs

BGP route target filtering (also known as route target constrain, or RTC) allows you to distribute VPN routes to only the devices that need them. In VPN networks without BGP route target filtering configured, BGP distributes all VPN routes to all VPN peer devices, which can strain network resources. The route target filtering feature was introduced to reduce the number of devices receiving VPN routes and VPN routing updates, thereby limiting the amount of overhead associated with running a VPN. The Junos OS implementation for BGP route target filtering is based on RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*.

What if you have a network environment where route target filtering is not widely deployed, or what if some devices do not support route target filtering? For example, you might have a BGP speaker with route target filtering enabled that is peered with a BGP speaker that does not support or have route target filtering configured. In this case, the BGP speaker with route target filtering configured must advertise default route target membership (RT membership) on behalf of its peer. The route target filtering resource savings are unrealized because the device supporting the filtering must now send all VPN routes to the device that does not support the filter. Proxy BGP route target filtering (or Proxy RTC) permits the generation of RT membership for devices that do not support route target filtering. This eases the deployment of route target filtering in networks where it is incompletely deployed or not fully supported.

Proxy BGP route target filtering allows you to distribute proxy RT membership advertisements created from the received BGP VPN routes to other devices in the network that need them. These are known as proxy advertisements because the device creates the RT membership on behalf of its peers without the route target filtering functionality. Proxy BGP route target filtering uses BGP route target extended communities that are exported to a specific BGP speaker to generate the route targets. Generated proxy RTC routes are stored in the bgp.rtarget.0 routing table.

You can also configure a policy to control which VPN routes are used to generate the proxy RTC routes. This can help control which RT membership is generated by the proxying device. In addition, you can configure a policy to reduce the memory overhead associated with proxy RTC. Proxy RTC only uses additional memory on a per-VPN route basis when it is permitted by a policy to be used for generating RT membership.

Example: Configuring Proxy BGP Route Target Filtering for VPNs

IN THIS SECTION

- Requirements | 60
- Overview | 61
- Configuration | 62
- Verification | 81

This example shows how to configure proxy BGP route target filtering (also known as proxy route target constrain, or proxy RTC).

Requirements

This example uses the following hardware and software components:

- Four Juniper Networks devices that can be a combination of M Series, MX Series, or T Series routers.
- Junos OS Release 12.2 or later on one or more devices configured for proxy BGP route filtering. In this example, you explicitly configure proxy BGP route filtering on the route reflectors.

Before configuring proxy BGP route target filtering, make sure that you are familiar with and understand the following concepts:

- "Layer 2 VPNs" on page 144
- *Understanding Layer 3 VPNs*
- *Understanding VPN-IPv4 Addresses and Route Distinguishers*
- *Configuring Policies for the VRF Table on PE Routers in VPNs*
- *Configuring BGP Route Target Filtering for VPNs*
- BGP extended communities

Overview

IN THIS SECTION

- Topology Diagram | 61

Route target filtering decreases the number of devices in a network that receive VPN routes that are not needed. Proxy BGP route target filtering allows networks to take advantage of route target filtering in locations where the feature is not currently supported. By configuring this feature, you can realize many of the same network resource savings that are available to you if your network fully supported BGP route target filtering.

To configure proxy BGP route target filtering, you include the `family route-target proxy-generate` statement on the devices that will distribute proxy route target membership (RT membership) advertisements for the devices that do not support BGP route target filtering. The proxy BGP route target filtering routes are then stored in the bgp.rtarget.0 routing table.

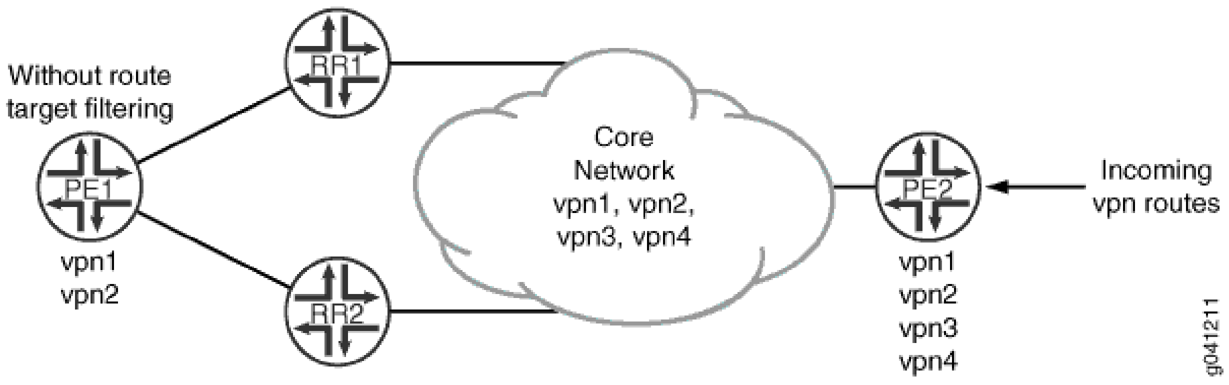
Proxy BGP route target filtering is intended to create RT membership advertisements for devices that do not support the BGP route target filtering feature. If the `proxy-generate` statement is present, but the route target family is negotiated with the BGP peer, the proxy-generate functionality is disabled. This allows simplified configuration of BGP peer groups where a portion of the peers in the group support route target filtering but others do not. In such an example case, the `family route-target proxy-generate` statement might be part of the BGP peer group configuration.

NOTE: When deploying proxy BGP route target filtering in your network, the `advertise-default` statement for BGP route target filtering causes the device to advertise the default route target route (0:0:0/0) and suppress all routes that are more specific. If you have proxy BGP route target filtering configured on one device and one or more peers have the `advertise-default` statement configured as part of their BGP route target filtering configuration, the advertise-default configuration is ignored.

Topology Diagram

Figure 5 on page 62 shows the topology used in this example.

Figure 5: Proxy BGP Route Target Filtering Topology


Without route target filtering
RR1
PE1
vpn1
vpn2
RR2
Core Network vpn1, vpn2, vpn3, vpn4
PE2
vpn1
vpn2
vpn3
vpn4
Incoming vpn routes
g041211

In this example, BGP route target filtering is configured on the route reflectors (Device RR1 and Device RR2) and the provider edge (PE) Device PE2, but the other PE, Device PE1, does not support the BGP route target filtering functionality. Device PE2 has four VPNs configured (vpn1, vpn2, vpn3, and vpn4). Device PE1 has two VPNs configured (vpn1 and vpn2), so this device is only interested in receiving route updates for vpn1 and vpn2. Currently, this is impossible because both route reflectors (Device RR1 and Device RR2) learn and share information about all of the incoming VPN routes (vpn1 through vpn4) with Device PE1. In the sample topology, all devices participate in autonomous system (AS) 203, OSPF is the configured interior gateway protocol (IGP), and LDP is the signaling protocol used by the VPNs. In this example, we use static routes in the VPN routing and forwarding (VRF) instances to generate VPN routes. This is done in place of using a PE to customer edge (CE) protocol such as OSPF or BGP.

To minimize the number of VPN route updates being processed by Device PE1, you include the `family route-target proxy-generate` statement to configure proxy BGP route target filtering on each route reflector. Each route reflector has a peering session with Device PE1 and supports route target filtering to the core. However, Device PE1 does not support route target filtering, so the network resource savings are unrealized by Device PE1 since it receives all of the VPN updates. By configuring proxy BGP route target filtering on the peering sessions facing Device PE1, you limit the number of VPN updates processed by Device PE1, and the route reflectors generate the proxy BGP route target routes for Device PE1 throughout the network.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 63
- Configuring Device PE1 | 66
- Configuring Device RR1 | 69
- Configuring Device RR2 | 73

- Configuring Device PE2 | 76

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 PE1

```
set interfaces ge-1/0/0 unit 0 description PE1-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE1-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.49.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.163.58
set protocols bgp group internal neighbor 10.255.165.220 family inet-vpn unicast
set protocols bgp group internal neighbor 10.255.165.28 family inet-vpn unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.163.58
set routing-options autonomous-system 203
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:203:100
set routing-instances vpn1 routing-options static route 203.0.113.1/24 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:203:101
set routing-instances vpn2 routing-options static route 203.0.113.2/24 discard
```

Device RR1

```
set interfaces ge-1/0/0 unit 0 description RR1-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR1-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.0.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 198.51.100.1
set protocols bgp group internal cluster 198.51.100.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.168.42 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.220
set routing-options autonomous-system 203
```

Device RR2

```
set interfaces ge-1/0/0 unit 0 description RR2-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.10.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR2-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.165.28
set protocols bgp group internal cluster 198.51.100.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.168.42 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.28
set routing-options autonomous-system 203
```

Device PE2

```
set interfaces ge-1/0/0 unit 0 description PE2-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.50.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE2-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.168.42
set protocols bgp group internal family inet-vpn unicast
set protocols bgp group internal family route-target
set protocols bgp group internal neighbor 10.255.165.220
set protocols bgp group internal neighbor 10.255.165.28
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.168.42
set routing-options autonomous-system 203
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:203:100
set routing-instances vpn1 routing-options static route 203.0.113.1/24 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:203:101
set routing-instances vpn2 routing-options static route 203.0.113.2/24 discard
set routing-instances vpn3 instance-type vrf
set routing-instances vpn3 vrf-target target:203:103
set routing-instances vpn3 routing-options static route 203.0.113.3/24 discard
set routing-instances vpn4 instance-type vrf
set routing-instances vpn4 vrf-target target:203:104
set routing-instances vpn4 routing-options static route 203.0.113.4/24 discard
```

Configuring Device PE1

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*.

To configure Device PE1:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-1/0/0 unit 0 description PE1-to-RR1
user@PE1# set ge-1/0/0 unit 0 family inet address 10.49.0.1/30
user@PE1# set ge-1/0/0 unit 0 family mpls
user@PE1# set ge-1/0/1 unit 0 description PE1-to-RR2
user@PE1# set ge-1/0/1 unit 0 family inet address 10.49.10.1/30
user@PE1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE1# set route-distinguisher-id 10.255.163.58
user@PE1# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE1# set type internal
user@PE1# set local-address 10.255.163.58
user@PE1# set neighbor 10.255.165.220 family inet-vpn unicast
user@PE1# set neighbor 10.255.165.28 family inet-vpn unicast
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
user@PE1# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:203:100
user@PE1# set routing-options static route 203.0.113.1/24 discard
```

```
[edit routing-instances vpn2]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:203:101
user@PE1# set routing-options static route 203.0.113.2/24 discard
```

7. 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 interfaces`, `show protocols`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show interfaces
ge-1/0/0 {
    unit 0 {
        description PE1-to-RR1;
        family inet {
            address 10.49.0.1/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description PE1-to-RR2;
        family inet {
            address 10.49.10.1/30;
        }
        family mpls;
    }
}
```

```
user@PE1# show protocols
bgp {
    group internal {
        type internal;
        local-address 10.255.163.58;
        neighbor 10.255.165.220 {
            family inet-vpn {
                unicast;
            }
        }
        neighbor 10.255.165.28 {
            family inet-vpn {
                unicast;
            }
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@PE1# show routing-options
route-distinguisher-id 10.255.14.182;
autonomous-system 203;
```

```
user@PE1# show routing-instances
vpn1 {
    instance-type vrf;
    vrf-target target:203:100;
    routing-options {
        static {
            route 203.0.113.1/24 discard;
        }
    }
}
vpn2 {
    instance-type vrf;
    vrf-target target:203:101;
    routing-options {
        static {
            route 203.0.113.2/24 discard;
        }
    }
}
```

Configuring Device RR1

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*.

To configure Device RR1:

1. Configure the interfaces.

```
[edit interfaces]
user@RR1# set ge-1/0/0 unit 0 description RR1-to-PE1
user@RR1# set ge-1/0/0 unit 0 family inet address 10.49.0.2/30
user@RR1# set ge-1/0/0 unit 0 family mpls
user@RR1# set ge-1/0/1 unit 0 description RR1-to-PE2
user@RR1# set ge-1/0/1 unit 0 family inet address 10.50.0.2/30
user@RR1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR1# set route-distinguisher-id 10.255.165.220
user@RR1# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR1# set type internal
user@RR1# set local-address 10.255.165.220
user@RR1# set cluster 198.51.100.1
user@RR1# set neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn unicast
user@RR1# set neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
user@RR1# set interface lo0.0 passive
```

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

```
[edit]
user@RR1# commit
```

Results

From configuration mode, confirm your configuration by entering the `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@RR1# show interfaces
ge-1/0/0 {
    unit 0 {
        description RR1-to-PE1;
        family inet {
            address 10.49.0.2/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description RR1-to-PE2;
        family inet {
            address 10.50.0.2/30;
        }
        family mpls;
    }
}
```

```
user@RR1# **show protocols**
bgp {
    group internal {
        type internal;
        local-address 198.51.100.1;
        cluster 198.51.100.1;
        neighbor 10.255.163.58 {
            description vpn1-to-pe1;
            family inet-vpn {
                unicast;
            }
            family route-target {
                proxy-generate;
            }
        }
        neighbor 10.255.168.42 {
            description vpn1-to-pe2;
            family inet-vpn {
                unicast;
            }
            family route-target;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@RR1# show routing-options
route-distinguisher-id 10.255.165.220;
autonomous-system 203;
```

Configuring Device RR2

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*.

To configure Device RR2:

1. Configure the interfaces.

```
[edit interfaces]
user@RR2# set ge-1/0/0 unit 0 description RR2-to-PE1
user@RR2# set ge-1/0/0 unit 0 family inet address 10.49.10.2/30
user@RR2# set ge-1/0/0 unit 0 family mpls
user@RR2# set ge-1/0/1 unit 0 description RR2-to-PE2
user@RR2# set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
user@RR2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR2# set route-distinguisher-id 10.255.165.28
user@RR2# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR2# set type internal
user@RR2# set local-address 10.255.165.28
user@RR2# set cluster 198.51.100.1
user@RR2# set neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn unicast
user@RR2# set neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
user@RR2# set interface lo0.0 passive
```

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

```
[edit]
user@RR2# commit
```

Results

From configuration mode, confirm your configuration by entering the `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@RR2# show interfaces
ge-1/0/0 {
    unit 0 {
        description RR2-to-PE1;
        family inet {
            address 10.49.10.2/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description RR2-to-PE2;
        family inet {
            address 10.50.10.2/30;
        }
        family mpls;
    }
}
```

```
user@RR2# show protocols
bgp {
    group internal {
        local-address 10.255.165.28;
        cluster 198.51.100.1;
        neighbor 10.255.163.58 {
            description vpn2-to-pe1;
            family inet-vpn {
                unicast;
            }
            family route-target {
                proxy-generate;
            }
        }
        neighbor 10.255.168.42 {
            description vpn2-to-pe2;
            family inet-vpn {
                unicast;
            }
            family route-target;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@RR2# show routing-options
route-distinguisher-id 10.255.165.28;
autonomous-system 203;
```

Configuring Device PE2

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*.

To configure Device PE2:

1. Configure the interfaces.

   ```
   [edit interfaces]
   user@PE2# set ge-1/0/0 unit 0 description PE2-to-RR1
   user@PE2# set ge-1/0/0 unit 0 family inet address 10.50.0.1/30
   user@PE2# set ge-1/0/0 unit 0 family mpls
   ```

```
user@PE2# set ge-1/0/1 unit 0 description PE2-to-RR2
user@PE2# set ge-1/0/1 unit 0 family inet address 10.50.10.1/30
user@PE2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE2# set route-distinguisher-id 10.255.168.42
user@PE2# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE2# set type internal
user@PE2# set local-address 10.255.168.42
user@PE2# set family inet-vpn unicast
user@PE2# set family route-target
user@PE2# set neighbor 10.255.165.220
user@PE2# set neighbor 10.255.165.28
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
user@PE2# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE2# set instance-type vrf
```

```
user@PE2# set vrf-target target:203:100
user@PE2# set routing-options static route 203.0.113.1/24 discard
```

```
[edit routing-instances vpn2]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:101
user@PE2# set routing-options static route 203.0.113.2/24 discard
```

```
[edit routing-instances vpn3]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:103
user@PE2# set routing-options static route 203.0.113.3/24 discard
```

```
[edit routing-instances vpn4]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:104
user@PE2# set routing-options static route 203.0.113.4/24 discard
```

7. 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 interfaces`, `show protocols`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE2# show interfaces
ge-1/0/0 {
    unit 0 {
        description PE2-to-RR1;
        family inet {
            address 10.50.0.1/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description PE2-to-RR2;
        family inet {
            address 10.50.10.1/30;
        }
        family mpls;
    }
}
```

```
user@PE2# show protocols
bgp {
    group internal {
        type internal;
        local-address 10.255.168.42;
        family inet-vpn {
            unicast;
        }
        family route-target;
        neighbor 10.255.165.220;
        neighbor 10.255.165.28;
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@PE2# show routing-options
route-distinguisher-id 10.255.168.42;
autonomous-system 203;
```

```
user@PE2# show routing-instances
vpn1 {
    instance-type vrf;
    vrf-target target:203:100;
    routing-options {
        static {
            route 203.0.113.1/24 discard;
        }
    }
}
vpn2 {
    instance-type vrf;
    vrf-target target:203:101;
    routing-options {
        static {
            route 203.0.113.2/24 discard;
        }
    }
}
vpn3 {
    instance-type vrf;
    vrf-target target:203:103;
    routing-options {
        static {
            route 203.0.113.3/24 discard;
        }
    }
}
vpn4 {
    instance-type vrf;
    vrf-target target:203:104;
    routing-options {
        static {
            route 203.0.113.4/24 discard;
        }
    }
}
```

Verification

IN THIS SECTION

● Verifying the Proxy BGP Route Target Routes | 81

Confirm that the configuration is working properly.

Verifying the Proxy BGP Route Target Routes

Purpose

Verify that the proxy BGP route target routes are displayed in the bgp.rtarget.0 table on Device RR1.

Action

From operational mode, enter the `show route table bgp.rtartget.0` command to display the proxy BGP route targets.

```
user@RR1# show route table bgp.rtarget.0
4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

203:203:100/96
                   *[RTarget/5] 00:01:22
                      Type Proxy
                        for 10.255.163.58
                         Local
                    [BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                      AS path: I, validation-state: unverified
                    > to 10.50.0.1 via ge-1/0/1
203:203:101/96
                   *[RTarget/5] 00:01:22
```

```
                        Type Proxy
                          for 10.255.163.58
                           Local
                      [BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                        AS path: I, validation-state: unverified
                      > to 10.50.0.1 via ge-1/0/1
203:203:103/96
                     *[BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                        AS path: I, validation-state: unverified
                      > to 10.50.0.1 via ge-1/0/1
203:203:104/96
                     *[BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                        AS path: I, validation-state: unverified
                      > to 10.50.0.1 via ge-1/0/1
```

Meaning

Device RR1 is generating the proxy BGP route target routes on behalf of its peer Device PE1. The proxy BGP route target routes are identified with the protocol and preference `[RTarget/5]` and the route target type of `Proxy`.

Example: Configuring an Export Policy for BGP Route Target Filtering for VPNs

IN THIS SECTION

- Requirements | 83
- Overview | 83
- Configuration | 85
- Verification | 105

This example shows how to configure an export routing policy for BGP route target filtering (also known as route target constrain, or RTC).

Requirements

This example uses the following hardware and software components:

- Four Juniper Networks devices that support BGP route target filtering.
- Junos OS Release 12.2 or later on one or more devices configured for proxy BGP route filtering. In this example, you explicitly configure proxy BGP route filtering on the route reflectors.

Before configuring an export policy for BGP route target filtering, make sure that you are familiar with and understand the following concepts:

- "Layer 2 VPNs" on page 144
- *Understanding Layer 3 VPNs*
- *Understanding VPN-IPv4 Addresses and Route Distinguishers*
- *Configuring Policies for the VRF Table on PE Routers in VPNs*
- *Configuring BGP Route Target Filtering for VPNs*
- BGP extended communities

Overview

IN THIS SECTION

- Topology Diagram | **84**

BGP route target filtering allows you to reduce network resource consumption by distributing route target membership (RT membership) advertisements throughout the network. BGP uses the RT membership information to send VPN routes only to the devices that need them in the network. Similar to other types of BGP reachability, you can apply a routing policy to route target filtering routes to influence the network. When route target filtering is configured, restricting the flow of route target filtering routes also restricts the VPN routes that might be attracted by this RT membership. Configuring this policy involves:

- Creating a filter that defines the list of route target prefixes.
- Creating a policy to select a subset of the route target filters to use for BGP route target filtering.

To define the list of route target prefixes:

- You configure the `rtf-prefix-list` statement at the `[edit policy-options]` hierarchy level to specify the name of the route target prefix list and one or more route target prefixes to use. This configuration allows you to specify the incoming route target filtering routes that the device will use and then distribute them throughout the network.

To configure the routing policy and apply the route target prefix list to that policy, you can specify the following policy options:

- `family route-target`—(Optional) The route-target family match condition specifies matching BGP route target filtering routes. You define this criteria in the `from` statement. This example shows how to create an export policy using the `family route-target` match condition.

  NOTE: Juniper uses the inet.3 table to resolve the next hop address when `family route-target` is configured.

- `protocol route-target`—(Optional) The route-target protocol match condition defines the criteria that an incoming route must match. You define this criteria in the `from` statement. This statement is primarily useful for restricting the policy to locally generated route target filtering routes.

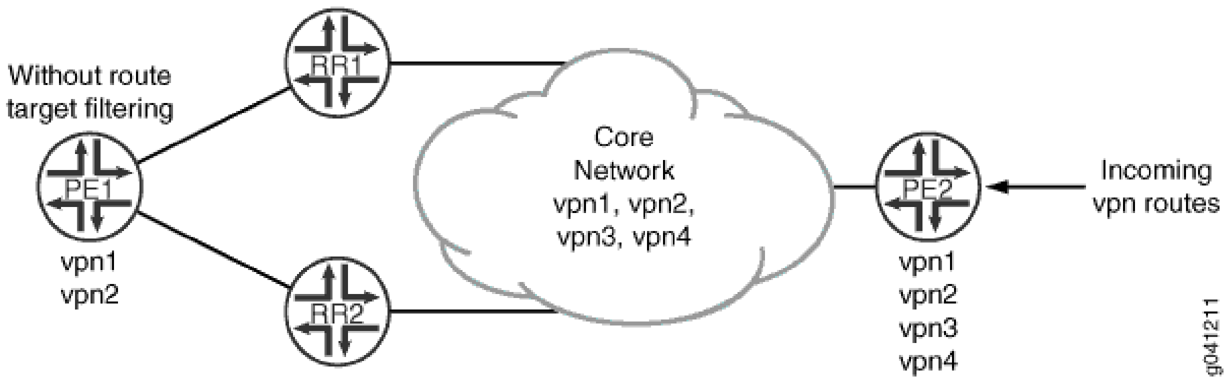
  NOTE: When you use the `show route table bgp.rtarget.0` command to view proxy BGP route target filtering routes, you will see the BGP protocol for received routes and the route target protocol routes for local route target filtering routes.

- `rtf-prefix-list` *name*—The rtf-prefix-list statement applies the list of route target prefixes that you already configured to the policy. You define this criteria in the `from` statement.

Topology Diagram

Figure 6 on page 85 shows the topology used in this example.

Figure 6: BGP Route Target Filtering Export Policy Topology



In this example, BGP route target filtering is configured on the route reflectors (Device RR1 and Device RR2) and provider edge (PE) Device PE2. The other PE, Device PE1, does not support BGP route target filtering. Proxy BGP route target filtering is also configured on the peering sessions between the route reflectors and Device PE1 to minimize the number of VPN route updates processed by Device PE1. Device PE2 has four VPNs configured (vpn1, vpn2, vpn3, and vpn4), and Device PE1 has two VPNs configured (vpn1 and vpn2). In the sample topology, all devices participate in autonomous system (AS) 203, OSPF is the configured interior gateway protocol (IGP), and LDP is the signaling protocol used by the VPNs. In this example, we use static routes in the VPN routing and forwarding (VRF) instances to generate VPN routes. This is done in place of using a PE to customer edge (CE) protocol such as OSPF or BGP.

In this example, you further control the routes being advertised from Device PE2 to Device PE1 by configuring an export policy on Device PE2 to prevent vpn3 routes from being advertised to Device RR1. You create a policy that specifies the `family route-target` match condition, defines the list of route target prefixes, and applies the list of route target prefixes by defining the `rtf-prefix-list` criteria.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 86
- Configuring Device PE1 | 89
- Configuring Device RR1 | 92
- Configuring Device RR2 | 96
- Configuring Device PE2 | 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 PE1

```
set interfaces ge-1/0/0 unit 0 description PE1-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE1-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.49.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.163.58
set protocols bgp group internal neighbor 10.255.165.220 family inet-vpn unicast
set protocols bgp group internal neighbor 10.255.165.28 family inet-vpn unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.163.58
set routing-options autonomous-system 203
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:203:100
set routing-instances vpn1 routing-options static route 203.0.113.1/24 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:203:101
set routing-instances vpn2 routing-options static route 203.0.113.2/24 discard
```

Device RR1

```
set interfaces ge-1/0/0 unit 0 description RR1-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR1-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.0.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 198.51.100.0
set protocols bgp group internal cluster 198.51.100.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn
unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.220
set routing-options autonomous-system 203
```

Device RR2

```
set interfaces ge-1/0/0 unit 0 description RR2-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.10.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR2-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.165.28
set protocols bgp group internal cluster 198.51.100.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn
unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.28
set routing-options autonomous-system 203
```

Device PE2

```
set interfaces ge-1/0/0 unit 0 description PE2-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.50.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE2-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.168.42
set protocols bgp group internal family inet-vpn unicast
set protocols bgp group internal family route-target
set protocols bgp group internal neighbor 10.255.165.220 export filter-rtc
set protocols bgp group internal neighbor 10.255.165.28
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options rtf-prefix-list exclude-103 203:203:103/96
set policy-options policy-statement filter-rtc from family route-target
set policy-options policy-statement filter-rtc from rtf-prefix-list exclude-103
set policy-options policy-statement filter-rtc then reject
set routing-options route-distinguisher-id 10.255.168.42
set routing-options autonomous-system 203
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:203:100
set routing-instances vpn1 routing-options static route 203.0.113.1/24 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:203:101
set routing-instances vpn2 routing-options static route 203.0.113.2/24 discard
set routing-instances vpn3 instance-type vrf
set routing-instances vpn3 vrf-target target:203:103
set routing-instances vpn3 routing-options static route 203.0.113.3/24 discard
set routing-instances vpn4 instance-type vrf
set routing-instances vpn4 vrf-target target:203:104
set routing-instances vpn4 routing-options static route 203.0.113.4/24 discard
```

Configuring Device PE1

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*.

To configure Device PE1:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-1/0/0 unit 0 description PE1-to-RR1
user@PE1# set ge-1/0/0 unit 0 family inet address 10.49.0.1/30
user@PE1# set ge-1/0/0 unit 0 family mpls
user@PE1#set ge-1/0/1 unit 0 description PE1-to-RR2
user@PE1#set ge-1/0/1 unit 0 family inet address 10.49.10.1/30
user@PE1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE1# set route-distinguisher-id 10.255.163.58
user@PE1# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE1# set type internal
user@PE1# set local-address 10.255.163.58
user@PE1# set neighbor 10.255.165.220 family inet-vpn unicast
user@PE1# set neighbor 10.255.165.28 family inet-vpn unicast
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
user@PE1# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:203:100
user@PE1# set routing-options static route 203.0.113.1/24 discard
```

```
[edit routing-instances vpn2]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:203:101
user@PE1# set routing-options static route 203.0.113.2/24 discard
```

7. 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 interfaces`, `show protocols`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show interfaces
ge-1/0/0 {
    unit 0 {
        description PE1-to-RR1;
        family inet {
            address 10.49.0.1/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description PE1-to-RR2;
        family inet {
            address 10.49.10.1/30;
        }
        family mpls;
    }
}
```

```
user@PE1# show protocols
bgp {
    group internal {
        type internal;
        local-address 10.255.163.58;
        neighbor 10.255.165.220 {
            family inet-vpn {
                unicast;
            }
        }
        neighbor 10.255.165.28 {
            family inet-vpn {
                unicast;
            }
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@PE1# show routing-options
route-distinguisher-id 10.255.14.182;
autonomous-system 203;
```

```
user@PE1# show routing-instances
vpn1 {
    instance-type vrf;
    vrf-target target:203:100;
    routing-options {
        static {
            route 203.0.113.1/24 discard;
        }
    }
}
vpn2 {
    instance-type vrf;
    vrf-target target:203:101;
    routing-options {
        static {
            route 203.0.113.2/24 discard;
        }
    }
}
```

Configuring Device RR1

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*.

To configure Device RR1:

1. Configure the interfaces.

```
[edit interfaces]
user@RR1# set ge-1/0/0 unit 0 description RR1-to-PE1
user@RR1# set ge-1/0/0 unit 0 family inet address 10.49.0.2/30
user@RR1# set ge-1/0/0 unit 0 family mpls
user@RR1# set ge-1/0/1 unit 0 description RR1-to-PE2
user@RR1# set ge-1/0/1 unit 0 family inet address 10.50.0.2/30
user@RR1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR1# set route-distinguisher-id 10.255.165.220
user@RR1# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR1# set type internal
user@RR1# set local-address 10.255.165.220
user@RR1# set cluster 198.51.100.1
user@RR1# set neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn unicast
user@RR1# set neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
user@RR1# set interface lo0.0 passive
```

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

```
[edit]
user@RR1# commit
```

Results

From configuration mode, confirm your configuration by entering the `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@RR1# show interfaces
ge-1/0/0 {
    unit 0 {
        description RR1-to-PE1;
        family inet {
            address 10.49.0.2/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description RR1-to-PE2;
        family inet {
            address 10.50.0.2/30;
        }
        family mpls;
    }
}
```

```
user@RR1# **show protocols**
bgp {
    group internal {
        type internal;
        local-address 198.51.100.0;
        cluster 198.51.100.1;
        neighbor 10.255.163.58 {
            description vpn1-to-pe1;
            family inet-vpn {
                unicast;
            }
            family route-target {
                proxy-generate;
            }
        }
        neighbor 10.255.168.42 {
            description vpn1-to-pe2;
            family inet-vpn {
                unicast;
            }
            family route-target;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@RR1# show routing-options
route-distinguisher-id 10.255.165.220;
autonomous-system 203;
```

Configuring Device RR2

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*.

To configure Device RR2:

1. Configure the interfaces.

    ```
    [edit interfaces]
    user@RR2# set ge-1/0/0 unit 0 description RR2-to-PE1
    user@RR2# set ge-1/0/0 unit 0 family inet address 10.49.10.2/30
    user@RR2# set ge-1/0/0 unit 0 family mpls
    user@RR2# set ge-1/0/1 unit 0 description RR2-to-PE2
    user@RR2# set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
    user@RR2# set ge-1/0/1 unit 0 family mpls
    ```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR2# set route-distinguisher-id 10.255.165.28
user@RR2# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR2# set type internal
user@RR2# set local-address 10.255.165.28
user@RR2# set cluster 198.51.100.1
user@RR2# set neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn unicast
user@RR2# set neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR2# set interface ge-1/0/0
```

```
user@RR2# set interface ge-1/0/1
user@RR2# set interface lo0.0 passive
```

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

```
[edit]
user@RR2# commit
```

Results

From configuration mode, confirm your configuration by entering the `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@RR2# show interfaces
ge-1/0/0 {
    unit 0 {
        description RR2-to-PE1;
        family inet {
            address 10.49.10.2/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description RR2-to-PE2;
        family inet {
            address 10.50.10.2/30;
        }
        family mpls;
    }
}
```

```
user@RR2# show protocols
bgp {
    group internal {
        local-address 10.255.165.28;
        cluster 198.51.100.1;
        neighbor 10.255.163.58 {
            description vpn2-to-pe1;
            family inet-vpn {
                unicast;
            }
            family route-target {
                proxy-generate;
            }
        }
        neighbor 10.255.168.42 {
            description vpn2-to-pe2;
            family inet-vpn {
                unicast;
            }
            family route-target;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@RR2# show routing-options
route-distinguisher-id 10.255.165.28;
autonomous-system 203;
```

Configuring Device PE2

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*.

To configure Device PE2:

1. Configure the interfaces.

```
[edit interfaces]
user@PE2# set ge-1/0/0 unit 0 description PE2-to-RR1
user@PE2# set ge-1/0/0 unit 0 family inet address 10.50.0.1/30
user@PE2# set ge-1/0/0 unit 0 family mpls
user@PE2#set ge-1/0/1 unit 0 description PE2-to-RR2
user@PE2#set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
user@PE2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE2# set route-distinguisher-id 10.255.168.42
user@PE2# set autonomous-system 203
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE2# set type internal
user@PE2# set local-address 10.255.168.42
user@PE2# set family inet-vpn unicast
user@PE2# set family route-target
```

```
user@PE2# set neighbor 10.255.165.220
user@PE2# set neighbor 10.255.165.28
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
user@PE2# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:100
user@PE2# set routing-options static route 203.0.113.1/24 discard
```

```
[edit routing-instances vpn2]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:101
user@PE2# set routing-options static route 203.0.113.2/24 discard
```

```
[edit routing-instances vpn3]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:103
user@PE2# set routing-options static route 203.0.113.3/24 discard
```

```
[edit routing-instances vpn4]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:203:104
user@PE2# set routing-options static route 203.0.113.4/24 discard
```

7. Configure and apply the export routing policy.

```
[edit policy-options]
user@PE2# set rtf-prefix-list exclude-103 203:203:103/96
```

```
[edit policy-options policy-statement filter-rtc]
user@PE2# set from family route-target
user@PE2# set from rtf-prefix-list exclude-103
user@PE2# set then reject
[edit protocols bgp group internal]
user@PE2# set neighbor 10.255.165.220 export filter-rtc
```

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 interfaces`, `show protocols`, `show policy-options`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE2# show interfaces
ge-1/0/0 {
    unit 0 {
        description PE2-to-RR1;
        family inet {
            address 10.50.0.1/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description PE2-to-RR2;
        family inet {
            address 10.50.10.2/30;
        }
        family mpls;
    }
}
```

```
user@PE2# show protocols
    bgp {
        group internal {
            type internal;
            local-address 10.255.168.42;
            family inet-vpn {
                unicast;
            }
            family route-target;
            neighbor 10.255.165.220 {
                export filter-rtc;
            }
            neighbor 10.255.165.28;
        }
    }
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@PE2# show routing-options
route-distinguisher-id 10.255.168.42;
autonomous-system 203;
```

```
user@PE2# show policy-options
policy-statement filter-rtc {
    from {
        family route-target;
        rtf-prefix-list exclude-103;
    }
    then reject;
}
rtf-prefix-list exclude-103 {
    203:203:103/96;
}
```

```
user@PE2# show routing-instances
vpn1 {
    instance-type vrf;
    vrf-target target:203:100;
    routing-options {
        static {
            route 203.0.113.1/24 discard;
        }
    }
}
vpn2 {
    instance-type vrf;
    vrf-target target:203:101;
    routing-options {
        static {
            route 203.0.113.2/24 discard;
        }
    }
}
vpn3 {
    instance-type vrf;
    vrf-target target:203:103;
    routing-options {
        static {
            route 203.0.113.3/24 discard;
        }
    }
}
vpn4 {
    instance-type vrf;
    vrf-target target:203:104;
    routing-options {
        static {
            route 203.0.113.4/24 discard;
        }
    }
}
```

Verification

IN THIS SECTION

- Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR1 | **105**
- Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR2 | **106**

Confirm that the configuration is working properly.

Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR1

Purpose

Verify that the route prefix for vpn3 is not in Device RR1's bgp.rtarget.0 table. Since an export policy on Device PE2 was applied to prevent the advertisement of vpn3 routes to Device RR1, Device RR1 should not receive those advertisements.

Action

From operational mode, enter the `show route advertising-protocol bgp 10.255.165.220 table bgp.rtarget.0` command.

```
user@PE2# show route advertising-protocol bgp 10.255.165.220 table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 11 routes
(4 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  203:203:100/96          *                    Self      100        I
  203:203:101/96          *                    Self      100        I
  203:203:104/96          *                    Self      100        I
```

Meaning

The bgp.rtartget.0 table does not display 203:203:103/96, which is the route prefix for vpn3. That means the export policy was applied correctly.

Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR2

Purpose

Verify that the route prefix for vpn3 is in Device RR2's bgp.rtarget.0 table. Since an export policy was not applied on Device PE2 to prevent the advertisement of vpn3 routes to Device RR2, Device RR2 should receive advertisements from all of the VPNs.

Action

From operational mode, enter the `show route advertising-protocol bgp 10.255.165.28 table bgp.rtarget.0` command.

```
user@PE2# show route advertising-protocol bgp 10.255.165.28 table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 11 routes (4 active, 0 holddown, 0 hidden)
(4 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  203:203:100/96          *                    Self      100        I
  203:203:101/96          *                    Self      100        I
  203:203:103/96          *                    Self      100        I
  203:203:104/96          *                    Self      100        I
```

Meaning

The bgp.rtartget.0 table displays the route prefixes for all of the VPNs.

Reducing Network Resource Use with Static Route Target Filtering for VPNs

The BGP VPN route target extended community (RFC 4360, *BGP Extended Communities Attribute*) is used to determine VPN membership. Static route target filtering helps to prevent resources from being consumed in portions of the network where the VPN routes are not needed due to the lack of member PE routers (RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label*

Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs). Routers can originate routes into the RT-Constrain protocol to indicate their interest in receiving VPN routes containing route targets that match the RT-Constrain NLRI.

Normally, for the RT-Constrain feature to function properly, it must be broadly deployed throughout a network. If this is not the case, the feature is less useful, because the RT-Constrain BGP speaker facing a non-RT-Constrain speaker must advertise a default RT-Constrain route to the other RT-Constrain speakers on behalf of the peer that does not support the feature. This effectively removes the resource saving benefits of the feature in portions of the network where it is not supported since a default RT-Constrain route causes the PE router and all intervening PE routers to need to receive all VPN routes.

The static RT-Constrain feature enables you to partially deploy the RT-Constrain feature in a network. The feature is enabled at a boundary in the network where RT-Constrain is configured. However, some BGP VPN peers do not support RT-Constrain, typically PE routers. The route targets of those PE routers must be statically configured on the router. These route targets are disseminated using the RT-Constrain protocol.

The proxy RT-Constrain feature permits BGP VPN peers that do not support the protocol to have their route-targets discovered and disseminated automatically. However, this feature can only support symmetric route-targets. For example, the import and export route-targets for a VRF routing instance are identical. However, for a hub-and-spoke VPN, the import and export route-targets are not identical. In this scenario, the import and export route-target may be statically configured to be disseminated in the RT-Constrain protocol.

CHAPTER 5

Configuring Forwarding Options for VPNs

IN THIS CHAPTER

- Chained Composite Next Hops for VPNs and Layer 2 Circuits | 108
- Example: Configuring Chained Composite Next Hops for Direct PE-PE Connections in VPNs | 109

Chained Composite Next Hops for VPNs and Layer 2 Circuits

IN THIS SECTION

- Benefits of chained composite next hops | 109

The Juniper Networks PTX Series Packet Transport Routers, MX Series 5G Universal Routing Platforms with MIC and MPC interfaces, and T4000 Core Routers are principally designed to handle large volumes of traffic in the core of large networks. Chained CNHs help to facilitate this capability by allowing the router to process much larger volumes of routes. A chained CNH allows the router to direct sets of routes sharing the same destination to a common forwarding next hop, rather than having each route also include the destination. In the event that a network destination is changed, rather than having to update all of the routes sharing that destination with the new information, only the shared forwarding next hop is updated with the new information. The chained CNHs continue to point to this forwarding next hop, which now contains the new destination.

When the next hops for MPLS LSPs are created on the routers, the tag information corresponding to the innermost MPLS label is extracted into a chained CNH. The chained CNH is stored in the ingress Packet Forwarding Engine. The chained CNH points to a next hop called the forwarding next hop that resides on the egress Packet Forwarding Engine. The forwarding next hop contains all the other information (all of the labels except for the inner-most labels as well as the IFA/IP information corresponding to the actual next-hop node). Many chained composite next hops can share the same forwarding next hop. Additionally, separating the inner-most label (that is the VPN label) from the forwarding next hop and storing it on the ingress PFE (within the chained composite next hop) helps to conserve egress Packet

Forwarding Engine memory by reducing the number of rewrite strings stored on the egress Packet Forwarding Engine.

Table 1 on page 109 shows support for chained CNHs for ingress or transit routers on the MPLS network.

Table 1: Support for Chained Composite Next Hops

Platform	L2 VPN	L3 VPN	L2 CKT
PTX Series	Ingress and transit	Ingress and transit	Ingress only
MX Series	Ingress only	Ingress only	Ingress only

To enable chained CNHs on a T4000 router, the chassis must be configured to use the `enhanced-mode` option in network services mode.

Benefits of chained composite next hops

Chained CNH optimizes the memory and performance of the router by reducing the size of the forwarding table. The router can use the same next-hop entry in the forwarding table for routes with different destinations when the next-hop is the same. This reduces the number of entries in the forwarding table and reduces the number of changes when the next hop entry has to be modified.

Example: Configuring Chained Composite Next Hops for Direct PE-PE Connections in VPNs

IN THIS SECTION

- Requirements | 110
- Overview and Topology | 110
- Configuration | 110

Requirements

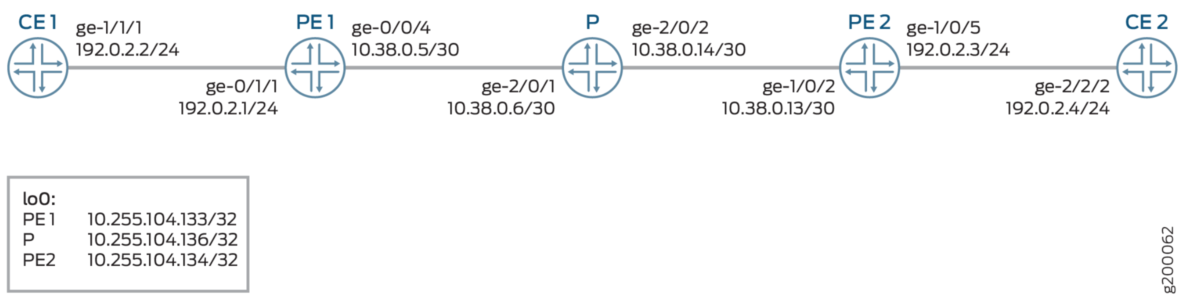
This example shows how to enable a Provider Edge (PE) router Layer 2 Virtual Private Network (VPN) connection with chained composite next hops for MIC and MPC interfaces on MX Series and T4000 routers. This example uses the following hardware and software components

- Five routers that can be a combination of MX240, MX480, MX960, or T4000 routers.
- Junos OS Release 17.3R1 or later running on all the devices.

Overview and Topology

Figure 7 on page 110 shows the sample topology of a Layer 2 VPN connection with chained composite next hops for MIC and MPC interfaces on MX series routers.

Figure 7: Chained Composite Next Hop on a PE Router



Configuration

IN THIS SECTION

- CLI Quick Configuration | 110
- Procedure | 113
- Results | 115

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.

CE1

```
set interfaces ge-1/1/1 unit 0 family inet address 192.0.2.2/24
set interfaces ge-1/1/1 unit 0 family iso
set interfaces ge-1/1/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 198.51.100.1/24
```

PE1

```
set interfaces ge-0/0/4 unit 0 family inet address 10.38.0.5/30
set interfaces ge-0/0/4 unit 0 family mpls
set interfaces ge-0/1/1 encapsulation ethernet-ccc
set interfaces ge-0/1/1  unit 0 family ccc
set interfaces lo0 unit 0 family inet address 10.255.104.133/32
set routing-options forwarding-table chained-composite-next-hop ingress l2vpn
set routing-options autonomous-system 200
set routing-options forwarding-table export lbpp
set protocols mpls interface ge-0/0/4.0
set protocols ospf area 0.0.0.0 interface ge-0/0/4.0
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.133
set protocols bgp group PEs family l2vpn signaling
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.134
set routing-instances vpn-a instance-type l2vpn
set routing-instances vpn-a interface ge-0/1/1.0
set routing-instances vpn-a route-distinguisher 200:1
set routing-instances vpn-a vrf-target target:200:1
set routing-instances vpn-a protocols l2vpn  encapsulation-type ethernet
 set routing-instances vpn-a protocols l2vpn site 100 site-identifier 100
set routing-instances vpn-a protocols l2vpn site 100  interface ge-0/1/1.0 remote-site-id 200
```

PE2

```
set interfaces ge-1/0/2 unit 0 family inet address 10.38.0.13/30
set interfaces ge-1/0/2 unit 0 family mpls
set interfaces ge-1/0/5 encapsulation ethernet-ccc
set interfaces ge-1/0/5 unit 0 family ccc
set interfaces lo0 unit 0 family inet address 10.255.104.134/32
set routing-options forwarding-table chained-composite-next-hop ingress l2vpn
set routing-options autonomous-system 200
set routing-options forwarding-table export lbpp
set protocols mpls interface ge-1/0/2.0
set protocols ospf area 0.0.0.0 interface ge-1/0/2.0
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.134
set protocols bgp group PEs family l2vpn signaling
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.133
set routing-instances vpn-a instance-type l2vpn
set routing-instances vpn-a interface ge-1/0/5.0
set routing-instances vpn-a route-distinguisher 200:1
set routing-instances vpn-a vrf-target target:200:1
set routing-instances vpn-a protocols l2vpn  encapsulation-type ethernet
set routing-instances vpn-a protocols l2vpn site 200 site-identifier 200
set routing-instances vpn-a protocols l2vpn site 200  interface ge-1/0/5.0 remote-site-id 100
```

P

```
set interfaces ge-2/0/1 unit 0 family inet address 10.38.0.6/30
set interfaces ge-2/0/1 unit 0 family mpls
set interfaces ge-2/0/2 unit 0 family inet address 10.38.0.14/30
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.104.136/32
set protocols mpls interface ge-2/0/1.0
set protocols ospf area 0.0.0.0 interface ge-2/0/1.0
set protocols mpls interface ge-2/0/2.0
set protocols ospf area 0.0.0.0 interface ge-2/0/2.0
set routing-options autonomous-system 200
```

CE2

```
set interfaces ge-2/2/2 unit 0 family inet address 192.0.2.4/24
set interfaces ge-2/2/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 198.51.100.2/24
```

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*.

To configure basic Layer 2 VPN with chained composite next hop on the PE1 router:

NOTE: Repeat this procedure for the PE2 router in the MPLS domain, after modifying the appropriate interface names, addresses, and any other parameters for the router.

1. Configure the interfaces on the PE1 router.

```
PE1 to CE1
[edit interfaces]
user@PE1# set interfaces ge-0/1/1 encapsulation ethernet-ccc
user@PE1# set interfaces ge-0/1/1  unit 0 family ccc
```

```
PE1 to P
[edit interfaces]
user@PE1 # set ge-0/0/4 unit 0 family inet address 10.38.0.5/30
user@PE1 # set ge-0/0/4 unit 0 family mpls
```

```
Loopback interface
[edit interfaces]
user@PE1 # set lo0 unit 0 family inet address 10.255.104.133/32
```

2. Enable chained composite next hop on the global Layer 2 VPN.

```
[edit routing-options]
use@PE1# set forwarding-table chained-composite-next-hop ingress l2vpn
```

3. Configure the autonomous system for PE1.

```
[edit routing-options]
user@PE1# set autonomous-system 200
```

4. Export the policy configured for load balancing.

```
[edit routing-options]
user@PE1# set forwarding-table export lbpp
```

5. Configure MPLS on the PE1 interfaces that connects to the P router.

```
[edit protocols]
set mpls interface ge-0/0/4.0
```

6. Configure OSPF on the PE1 nterface.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface ge-0/0/4.0
```

7. Configure the IBGP group for PE1 to PE2 router.

```
[edit protocols]
user@PE1# set bgp group PEs type internal
user@PE1# set bgp group PEs local-address 10.255.104.133
user@PE1# set bgp group PEs family l2vpn signaling
user@PE1# set bgp group PEs family inet-vpn unicast
user@PE1# set bgp group PEs neighbor 10.255.104.134
```

8. Configure the routing instance parameters.

```
[edit routing-instances]
user@PE1# set vpn-a instance-type l2vpn
user@PE1# set vpn-a interface ge-0/1/1.0
user@PE1# set vpn-a route-distinguisher 200:1
user@PE1# set vpn-a vrf-target target:200:1
user@PE1# set vpn-a protocols l2vpn  encapsulation-type ethernet
```

```
user@PE1# set vpn-a protocols l2vpn site 100 site-identifier 100
user@PE1# set vpn-a protocols l2vpn site 100  interface ge-0/1/1.0 remote-site-id 200
```

Results

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

PE1

```
user@PE1# show interfaces
ge-0/0/4 {
    unit 0 {
        family inet {
            address 10.38.0.5/30;
        }
        family mpls;
    }
}
ge-0/1/1 {
    encapsulation ethernet-ccc;
    unit 0 {
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.104.133/32;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface ge0/0/4.0;
}
```

```
bgp {
    group PEs {
        type internal;
        local-address 10.255.104.133;
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        neighbor 10.255.104.134;
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/4.0;
    }
}
```

```
user@PE1# show routing-options
autonomous-system 200;
forwarding-table {
    export lbpp;
    chained-composite-next-hop {
        ingress {
            l2vpn;
        }
    }
}
```

```
user@PE1# show routing-instances
vpn-a {
    instance-type l2vpn;
    interface ge-0/1/1.0;
    route-distinguisher 200:1;
    vrf-target target:200:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site 100 {
                site-identifier 100;
                interface ge-0/1/1.0 {
                    remote-site-id 200;
                }
            }
        }
    }
}
```

CHAPTER 6

Configuring Graceful Restart for VPNs

IN THIS CHAPTER

- VPN Graceful Restart | 118
- Configuring Graceful Restart for VPNs | 119
- Enabling Unicast Reverse-Path Forwarding Check for VPNs | 122
- Understanding and Preventing Unknown Unicast Forwarding | 123

VPN Graceful Restart

IN THIS SECTION

- Benefit of a VPN graceful restart | 119

With routing protocols, any service interruption requires that an affected router recalculate adjacencies with neighboring routers, restore routing table entries, and update other protocol-specific information. An unprotected restart of the router results in forwarding delays, route flapping, wait times stemming from protocol reconvergence, and even dropped packets. Graceful restart allows a routing device undergoing a restart to inform its adjacent neighbors and peers of its condition. During a graceful restart, the restarting device and its neighbors continue forwarding packets without disrupting network performance.

For VPN graceful restart to function properly, the following items need to be configured on the PE router:

- BGP graceful restart must be active on the PE-to-PE sessions carrying any service-signaling data in the session’s network layer reachability information (NLRI).
- OSPF, IS-IS, LDP, and RSVP graceful restart must be active, because routes added by these protocols are used to resolve VPN NLRIs.

- For other protocols (static, Routing Information Protocol [RIP], and so on), graceful restart functionality must also be active when these protocols are run between the PE and CE routers. Layer 2 VPNs do not rely on this, because protocols are not configured between the PE and CE routers.

In VPN graceful restart, a restarting router completes the following procedures:

- Waits for all the BGP NLRI information from other PE routers before it starts advertising routes to its CE routers.
- Waits for all protocols in all routing instances to converge (or finish graceful restart) before sending CE router information to the other PE routers.
- Waits for all routing instance information (whether it is local configuration or advertisements from a remote peer router) to be processed before sending it to the other PE routers.
- Preserves all forwarding state information in the MPLS routing tables until new labels and transit routes are allocated and then advertises them to other PE routers (and CE routers in carrier-of-carriers VPNs).

Graceful restart is supported on Layer 2 VPNs, Layer 3 VPNs, and virtual-router routing instances.

Benefit of a VPN graceful restart

The main benefit of a VPN graceful restart is that it allows a router whose VPN control plane is undergoing a restart to continue to forward traffic while recovering its state from neighboring routers. It temporarily suppresses all routing protocol updates and enables a router to pass through intermediate convergence states that are hidden from the rest of the network. Without graceful restart, a control plane restart disrupts the VPN services provided by the router.

Configuring Graceful Restart for VPNs

You can configure graceful restart to enable a router to pass through intermediate convergence states that are hidden from the rest of the network. Graceful restart allows a router whose VPN control plane is undergoing a restart (restarting router) to continue to forward traffic while recovering its state from neighboring routers (helper routers).

The restarting router requests a grace period from the neighbor or peer, which can then cooperate with the restarting router. When a restart event occurs and graceful restart is enabled, the restarting router can still forward traffic during the restart period, and convergence in the network is not disrupted. The helper routers hide the restart event from other devices not directly connected to the restarting router. In other words, the restart is not visible to the rest of the network, and the restarting router is not removed from the network topology.

Without graceful restart, a control plane restart disrupts any VPN services provided by the router. Graceful restart is supported on Layer 2 VPNs, Layer 3 VPNs, virtual-router routing instances, and VPLS.

The graceful restart request occurs only if the following conditions are met:

- The network topology is stable.
- The neighbor or peer routers cooperate.
- The restarting router is not already cooperating with another restart already in progress.
- The grace period does not expire.

Before you begin:

- Configure the devices for network communication.
- Configure the device interfaces.

Graceful restart is disabled by default. To enable VPN graceful restart:

1. Configure graceful restart globally.

```
[edit routing-options]
user@host# set graceful-restart
```

NOTE:

- Graceful restart can be enabled on logical systems. To configure graceful restart globally, include the `graceful-restart` statement at the `[edit logical-systems` *`logical-system-name`* `routing-options]` or the `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `routing-options]` hierarchy levels.
- To disable graceful restart globally, include the `disable` statement at the `[edit routing-options graceful-restart]` hierarchy level.

  For example:

  ```
  [edit routing-options]
  user@host# set graceful-restart disable
  ```

2. Enable or disable graceful restart on a per-protocol, per-group, or per-neighbor basis, depending on the specific protocol, where the most specific definition is used.

```
[edit protocols]
user@host# set bgp graceful-restart
user@host# set bgp group group-name type internal local-address local-ip-address neighbor
neighbor1-address
user@host# set bgp group group-name type internal local-address local-ip-address neighbor
neighbor2-address graceful-restart disable
```

3. Configure graceful restart for Layer 3 VPNS for all routing and MPLS-related protocols within a routing instance. Because you can configure multi-instance BGP and multi-instance LDP, graceful restart for a carrier-of-carriers scenario is supported.

```
[edit routing-instance]
user@host# set routing-instance-name routing-options graceful-restart
```

NOTE:

- To disable graceful restart globally, include the `disable` statement at the `[edit routing-instances` *`routing-instance-name`* `routing-options graceful-restart]` hierarchy level.

  For example:

  ```
  [edit routing-instances]
  user@host# set instance1 routing-options graceful-restart disable
  ```

- To disable graceful restart for individual protocols, include the `disable` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols` *`protocol-name`* `graceful-restart]` hierarchy level.

  For example:

  ```
  [edit routing-instances]
  user@host# set instance1 protocols ospf graceful-restart disable
  ```

4. Configure the duration of the graceful restart period for the routing instance.

   ```
   [edit routing-options]
   user@host# set graceful-restart restart-duration seconds
   ```

   The `restart-duration` option sets the period of time that the router waits for a graceful restart to be completed. You can configure a time between 1 through 600 seconds. The default value is 300 seconds. At the end of the configured time period, the router performs a standard restart without recovering its state from the neighboring routers. This disrupts VPN services, but is probably necessary if the router is not functioning normally.

   NOTE: You can include the `restart-duration` option at either the global or routing instance level.

Enabling Unicast Reverse-Path Forwarding Check for VPNs

IP spoofing may occur during a denial-of-service (DoS) attack. IP spoofing allows an intruder to pass IP packets to a destination as genuine traffic, when in fact the packets are not actually meant for the destination. This type of spoofing is harmful because it consumes the destination's resources.

Unicast reverse-path forwarding (RPF) check is a tool to reduce forwarding of IP packets that may be spoofing an address. A unicast RPF check performs a route table lookup on an IP packet's source address, and checks the incoming interface. The router determines whether the packet is arriving from a path that the sender would use to reach the destination. If the packet is from a valid path, the router forwards the packet to the destination address. If it is not from a valid path, the router discards the packet. Unicast RPF is supported for the IPv4 and IPv6 protocol families, as well as for the virtual private network (VPN) address family. You can also enable unicast RPF within a VPN routing instance.

To enable unicast RPF check, include the `unicast-reverse-path` statement:

```
unicast-reverse-path (active-paths | feasible-paths);
```

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

To consider only active paths during the unicast RPF check, include the `active-paths` option. To consider all feasible paths during the unicast RPF check, include the `feasible-paths` option.

For more information about how to configure the `unicast-reverse-path` statement, see *Example: Configuring Unicast RPF (On a Router)* and .

RELATED DOCUMENTATION

Example: Configuring Unicast RPF (On a Router)

Understanding and Preventing Unknown Unicast Forwarding

IN THIS SECTION

- Verifying That Unknown Unicast Packets Are Forwarded to a Single Interface | **124**
- Configuring Unknown Unicast Forwarding (ELS) | **125**
- Verifying That Unknown Unicast Packets Are Forwarded to a Trunk Interface | **128**
- Configuring Unknown Unicast Forwarding (CLI Procedure) | **130**
- Configuring Unknown Unicast Drop (CLI Procedure) | **130**
- Platform-Specific Configuring Unknown Unicast Forwarding Behavior | **132**

Unknown unicast traffic consists of unicast packets with unknown destination MAC addresses. By default, the switch floods these unicast packets that traverse a VLAN to all interfaces that are members of that VLAN. Forwarding this type of traffic can create unnecessary traffic that leads to poor network performance or even a complete loss of network service. This flooding of packets is known as a traffic storm.

To prevent a traffic storm, you can disable the flooding of unknown unicast packets to all VLAN interfaces by configuring specific VLANs or all VLANs to forward all unknown unicast traffic traversing them to a specific interface. You can configure multiple VLANs to forward unknown unicast packets to the same interface or configure different interfaces for different VLANs. This channels the unknown unicast traffic traversing VLANs to specific interfaces instead of flooding all interfaces.

NOTE: The unknown-unicast-forwarding feature is not supported on QFX10000 Series platforms.

Verifying That Unknown Unicast Packets Are Forwarded to a Single Interface

IN THIS SECTION

- Purpose | 124
- Action | 124
- Meaning | 125

Purpose

Verify that a VLAN is forwarding all unknown unicast packets (those with unknown destination MAC addresses) to a single interface instead of flooding unknown unicast packets across all interfaces that are members of that VLAN.

NOTE: This procedure uses Junos OS for EX Series switches with support for the Enhanced Layer 2 Software (ELS) configuration style. For ELS details see: Using the Enhanced Layer 2 Software CLI.

Action

(EX4300 Switches) Display the forwarding interface for unknown unicast packets for a VLAN (here, the VLAN name is v1):

```
user@switch> show configuration switch-options

unknown-unicast-forwarding {
    vlan v1 {
        interface ge-0/0/7.0;
    }
}
```

(EX9200 Switches) Display the forwarding interface for unknown unicast packets:

```
user@switch> show forwarding-options

next-hop-group uuf-nhg {
    group-type layer-2;
    interface ge-0/0/7.0;
}
```

Meaning

The sample output from the `show` commands show that the unknown unicast forwarding interface for VLAN `v1`is interface `ge-0/0/7`.

Configuring Unknown Unicast Forwarding (ELS)

IN THIS SECTION

- Configuring Unknown Unicast Forwarding on EX4300 Switches | **125**
- Configuring Unknown Unicast Forwarding on EX9200 Switches | **126**

NOTE: This task uses Junos OS for EX Series switches or QFX Series with support for the Enhanced Layer 2 Software (ELS) configuration style. For ELS details, see Using the Enhanced Layer 2 Software CLI

The unknown-unicast-forwarding feature is not supported on QFX10000 Series platforms.

Unknown unicast traffic consists of packets with unknown destination MAC addresses. By default, the switch floods these packets that traverse a VLAN to all interfaces associated with that VLAN. This flooding of packets is known as a traffic storm and can negatively impact network performance.

To prevent flooding unknown unicast traffic across the switch, configure unknown unicast forwarding to direct all unknown unicast packets within a VLAN to a specific interface. You can configure each VLAN to divert unknown unicast traffic to a different interface or use the same interface for multiple VLANs.

Configuring Unknown Unicast Forwarding on EX4300 Switches

To configure unknown unicast forwarding options on EX4300 switches:

- Configure unknown unicast forwarding for a specific VLAN and specify the interface to which all unknown unicast traffic will be forwarded:

  ```
  [edit switch-options]
  user@switch# set unknown-unicast-forwarding vlan vlan-name interface interface-name
  ```

- Configure unknown unicast forwarding for all VLANs and specify the interface to which all unknown unicast traffic will be forwarded:

  ```
  [edit switch-options]
  user@switch# set unknown-unicast-forwarding vlan all interface interface-name
  ```

Configuring Unknown Unicast Forwarding on EX9200 Switches

To configure unknown unicast forwarding on EX9200 switches, you must configure a flood filter and apply it to VLANs for which you want to configure unknown unicast forwarding. Flood filters are firewall filters that are applied only to broadcast, unknown unicast, and multicast (BUM) traffic. If a flood filter is configured, only traffic packets that are of the packet type `unknown-unicast` are forwarded to the interface on which unicast forwarding is configured. A next-hop group redirects the packets according to the action specified in the flood filter.

To configure the next-hop group that receives Layer 2 packets and then configure the interface to which these packets are forwarded:

1. Configure the `next-hop-group` action for the Layer 2 interface expected to receive unknown unicast packets:

   ```
   [edit forwarding-options]
   user@switch# set next-hop-group next-hop-group-name group-type layer-2
   [edit forwarding-options]
   user@switch# set next-hop-group next-hop-group-name interface interface-name
   ```

   For example:

   ```
   [edit forwarding-options]
   user@switch# set next-hop-group uuf-nhg group-type layer-2
   [edit forwarding-options]
   user@switch# set next-hop-group uuf-nhg interface ge-3/1/7.0
   ```

2. Configure a firewall filter with family address type `ethernet-switching`:

   ```
   [edit firewall]
   user@switch# set family ethernet-switching filter filter-name
   ```

   For example:

   ```
   [edit firewall]
   user@switch# set family ethernet-switching filter uuf_filter
   ```

3. Configure a term in the firewall filter for the interface that receives unknown unicast packets (the interface specified in Step 1) to discard unknown unicast packets:

   ```
   [edit firewall family ethernet-switching filter filter-name]
   user@switch# set term term-name from interface interface-name
   user@switch# set term term-name from traffic-type unknown-unicast
   user@switch# set term term-name then discard
   ```

   For example:

   ```
   [edit firewall family ethernet-switching filter uuf_filter]
   user@switch# set term source-drop from interface ge-3/1/7.0
   user@switch# set term source-drop from traffic-type unknown-unicast
   user@switch# set term source-drop then discard
   ```

4. Configure a term in the firewall filter for unknown unicast packets to be flooded to the interface enabled for unknown unicast forwarding by using `next-hop-group` (in step 1):

   ```
   [edit firewall family ethernet-switching filter filter-name]

   user@switch# set term term-name from traffic-type unknown-unicast
   user@switch# set term term-name then next-hop-group group-name
   ```

   For example:

   ```
   [edit firewall family ethernet-switching filter uuf_filter]
   ```

```
user@switch# set term uuf-flood from traffic-type unknown-unicast
user@switch# set term uuf-flood then next-hop-group uuf-nhg
```

5. Configure a default term for the firewall filter to forward packets other than unknown unicast packets:

```
[edit firewall family ethernet-switching filter filter-name]

user@switch# set term term-name then accept
```

For example:

```
[edit firewall family ethernet-switching filter uuf_filter]

user@switch# set term fwd-default then accept
```

6. Apply the filter as a flood filter on the VLAN that includes the interface which will receive unknown unicast packets:

```
[edit vlans vlan-name]

user@switch# set forwarding-options flood input filter-name
```

For example:

```
[edit vlans v1]

user@switch# set forwarding-options flood input uuf_filter
```

Verifying That Unknown Unicast Packets Are Forwarded to a Trunk Interface

IN THIS SECTION

- Purpose | 129
- Action | 129
- Meaning | 129

Purpose

Verify that a VLAN is forwarding all unknown unicast packets (those with unknown destination MAC addresses) to a single trunk interface instead of flooding unknown unicast packets across all interfaces that are members of the same VLAN.

Action

Display the forwarding interface for unknown unicast packets for a VLAN (here, the VLAN name is **v1**):

```
user@switch> show configuration ethernet-switching-options

unknown-unicast-forwarding {
    vlan v1 {
        interface ge-0/0/7.0;
    }
}
```

Display the Ethernet switching table:

```
user@switch> show ethernet-switching table vlan v1
Ethernet-switching table: 3 unicast entries
  VLAN              MAC address       Type         Age Interfaces
  v1                *                 Flood          - All-members
  v1                00:01:09:00:00:00 Learn         24 ge-0/0/7.0
  v1                00:11:09:00:01:00 Learn         37 ge-0/0/3.0
```

Meaning

The sample output from the `show configuration ethernet-switching-options` command shows that the unknown unicast forwarding interface for VLAN **v1** is interface **ge-0/0/7**. The `show ethernet-switching table` command shows that an unknown unicast packet is received on interface **ge-0/0/3** with the destination MAC address (DMAC) **00:01:09:00:00:00** and the source MAC address (SMAC) of **00:11:09:00:01:00**. This shows that the SMAC of the packet is learned in the normal way (through the interface **ge-0/0/3.0**), while the DMAC is learned on interface **ge-0/0/7**.

Configuring Unknown Unicast Forwarding (CLI Procedure)

Unknown unicast traffic consists of packets with unknown destination MAC addresses. By default, the switch floods these packets to all interfaces associated with a VLAN. Forwarding such traffic to interfaces on the switch can create a security issue.

To prevent flooding unknown unicast traffic across the switch, configure unknown unicast forwarding to direct all unknown unicast packets within a VLAN out to a specific trunk interface. From there, the destination MAC address can be learned and added to the Ethernet switching table. You can configure each VLAN to divert unknown unicast traffic to different trunk interfaces or use one trunk interface for multiple VLANs.

NOTE: For Junos OS for EX Series switches or QFX Series with support for the Enhanced Layer 2 Software (ELS) configuration style, see *Configuring Unknown Unicast Forwarding (ELS)*.

The unknown-unicast-forwarding feature is not supported on QFX10000 Series platforms.

To configure unknown unicast forwarding options:

NOTE: Before you can configure unknown unicast forwarding within a VLAN, you must first configure that VLAN.

1. Configure unknown unicast forwarding for a specific VLAN (here, the VLAN name is **employee**):

```
[edit ethernet-switching-options]
user@switch# set unknown-unicast-forwarding vlan employee
```

2. Specify the trunk interface to which all unknown unicast traffic will be forwarded:

```
[edit ethernet-switching-options]
user@switch# set unknown-unicast-forwarding vlan employee interface ge-0/0/3.0
```

Configuring Unknown Unicast Drop (CLI Procedure)

You can enhance network performance and prevent traffic storms by configuring your switch to drop unknown unicast packets. This action prevents the flooding of unicast packets with unknown destination MAC addresses across VLAN interfaces.

Unknown unicast drop should be configured at VLAN-level. When enabled on a VLAN, it is applied to all ports configured with that VLAN.

When you enable this feature, packets with unknown destination MAC addresses are dropped. However, the switch learns the source MAC address of these packets and adds it to the MAC address table. This approach ensures efficient network resources usage and optimal network performance.

To configure unknown unicast drop options:

NOTE: Before you can configure unknown unicast forwarding within a VLAN, you must first configure that VLAN.

1. Configure unknown unicast forwarding for a specific VLAN (here, the VLAN name is **employee**):

```
[edit ethernet-switching-options]
user@switch# set unknown-unicast-forwarding vlan employee
```

2. Specify the trunk interface to which all unknown unicast traffic will be forwarded:
3. Enable discarding of packets by issuing `drop` at the end of the CLI:

   NOTE: By default, the discard feature is disabled.

```
[edit ethernet-switching-options]
user@switch# set unknown-unicast-forwarding vlan employee interface ge-0/0/3.0
```

```
[edit ethernet-switching-options]
user@switch# set unknown-unicast-forwarding vlan employee drop
```

To **disable** the discarding of packets, issue the following command:

```
[edit ethernet-switching-options]
user@switch# delete unknown-unicast-forwarding vlan employee drop
```

NOTE:

- This feature is available on *Junos Evolved OS* only for QFX5130-32CD, QFX5130E-32CD, QFX5130-48C, QFX5130-48CM, QFX5700, and QFX5700E switches.

- On *Junos OS*, this feature is available only for QFX5120-32C, QFX5120-48T, QFX5120-48Y, QFX5120-48YM switches.

Platform-Specific Configuring Unknown Unicast Forwarding Behavior

Table 2: Platform-Specific Behavior

Platform	Difference
Junos Evolved ACX7000 series	ACX7000 series routers do not support configuring unknown unicast forwarding.

RELATED DOCUMENTATION

Understanding Storm Control

Configuring Autorecovery for Port Security Events

CHAPTER 7

Configuring Class of Service for VPNs

IN THIS CHAPTER

- VPNs and Class of Service | **133**
- Rewriting Class of Service Markers and VPNs | **133**

VPNs and Class of Service

You can configure Junos class-of-service (CoS) features to provide multiple classes of service for VPNs. The CoS features are supported on Layer2 VPNs, Layer 3 VPNs, and VPLS. On the router, you can configure multiple forwarding classes for transmitting packets, define which packets are placed into each output queue, schedule the transmission service level for each queue, and manage congestion using a random early detection (RED) algorithm.

VPNs use the standard CoS configuration.

Rewriting Class of Service Markers and VPNs

A marker reads the current forwarding class and loss priority information associated with a packet and finds the chosen code point from a table. It then writes the code point information into the packet header. Entries in a marker configuration represent the mapping of the current forwarding class into a new forwarding class, to be written into the header.

You define markers in the *rewrite rules* section of the class-of-service (CoS) configuration hierarchy and reference them in the *logical interface* configuration. You can configure different rewrite rules to handle VPN traffic and non-VPN traffic. The rewrite rule can be applied to MPLS and IPv4 packet headers simultaneously, making it possible to initialize MPLS experimental (EXP) and IP precedence bits at LSP ingress.

For a detailed example of how to configure rewrite rules for MPLS and IPv4 packets and for more information about how to configure statements at the `[edit class-of-service]` hierarchy level, see the Class of Service User Guide (Routers and EX9200 Switches).

CHAPTER 8

Pinging VPNs

IN THIS CHAPTER

- Pinging VPNs, VPLS, and Layer 2 Circuits | **134**
- Setting the Forwarding Class of the Ping Packets | **135**
- Pinging a VPLS Routing Instance | **135**
- Pinging a Layer 2 VPN | **136**
- Pinging a Layer 3 VPN | **137**
- Pinging a Layer 2 Circuit | **137**
- Pinging Customer Edge Device IP Address | **137**

Pinging VPNs, VPLS, and Layer 2 Circuits

For testing purposes, you can ping Layer 2 VPNs, Layer 3 VPNs, and Layer 2 circuits by using the `ping mpls` command. The `ping mpls` command helps to verify that a VPN or circuit has been enabled and tests the integrity of the VPN or Layer 2 circuit connection between the PE routers. It does not test the connection between a PE router and a CE router. To ping a VPLS routing instance, you issue a `ping vpls instance` command (see *Pinging a VPLS Routing Instance*).

You issue the `ping mpls` command from the ingress PE router of the VPN or Layer 2 circuit to the egress PE router of the same VPN or Layer 2 circuit. When you execute the `ping mpls` command, echo requests are sent as MPLS packets.

The payload is a User Datagram Protocol (UDP) packet forwarded to the address `127.0.0.1`. The contents of this packet are defined in RFC 4379, *Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures*. The label and interface information for building and sending this information as an MPLS packet is the same as for standard VPN traffic, but the time-to-live (TTL) of the innermost label is set to 1.

When the echo request arrives at the egress PE router, the contents of the packet are checked, and then a reply that contains the correct return is sent by means of UDP. The PE router sending the echo request waits to receive an echo reply after a timeout of 2 seconds (you cannot configure this value).

You must configure MPLS at the `[edit protocols mpls]` hierarchy level on the egress PE router (the router receiving the MPLS echo packets) to be able to ping the VPN or Layer 2 circuit. You must also configure the address `127.0.0.1/32` on the egress PE router's `lo0` interface. If this is not configured, the egress PE router does not have this forwarding entry and therefore simply drops the incoming MPLS pings.

The `ping mpls` command has the following limitations:

- You cannot ping an IPv6 destination prefix.
- You cannot ping a VPN or Layer 2 circuit from a router that is attempting a graceful restart.
- You cannot ping a VPN or Layer 2 circuit from a logical system.

You can also determine whether an LSP linking two PE routers in a VPN is up by pinging the end point address of the LSP. The command you use to ping an MPLS LSP end point is `ping mpls lsp-end-point` *`address`*. This command tells you what type of LSP (RSVP or LDP) terminates at the address specified and whether that LSP is up or down.

For a detailed description of this command, see the *Junos Routing Protocols and Policies Command Reference*.

Setting the Forwarding Class of the Ping Packets

When you execute the `ping mpls` command, the ping packets forwarded to the destination include MPLS labels. It is possible to set the value of the forwarding class for these ping packets by using the **exp** option with the `ping mpls` command. For example, to set the forwarding class to 5 when pinging a Layer 3 VPN, issue the following command:

```
ping mpls l3vpn westcoast source 192.0.2.0 prefix 192.0.2.1 exp 5 count 20 detail
```

This command would makes the router attempt to ping the Layer 3 VPN `westcoast` using ping packets with an EXP forwarding class of 5. The default forwarding class used for the `ping mpls` command packets is 7.

Pinging a VPLS Routing Instance

To ping a VPLS routing instance, use the following command:

```
ping vpls instance instance-name destination-mac address source-ip address <count number> <data-
plane-response> <detail> <learning-vlan-id number> <logical-system logical-system-name>
```

Pinging a VPLS routing instance requires using the `ping vpls instance` command with a combination of the routing instance name, the destination MAC address, and the source IP address (IP address of the outgoing interface).

When you run this command, you are provided feedback on the status of your request. An exclamation point (!) indicates that an echo reply was received. A period (`.`) indicates that an echo reply was not received within the timeout period. An `x` indicates that an echo reply was received with an error code these packets are not counted in the received packets count. They are accounted for separately.

Use Feature Explorer to confirm platform and release support for `ping vpls instance`. For more details, including argument descriptions and additional options, see ping vpls instance.

Pinging a Layer 2 VPN

To ping a Layer 2 VPN, use one of the following commands:

- `ping mpls l2vpn interface` *`interface-name`*

  You ping an interface configured for the Layer 2 VPN on the egress PE router.

- `ping mpls l2vpn instance` *`l2vpn-instance-name`* `local-site-id` *`local-site-id-number`* `remote-site-id` *`remote-site-id-number`*

  You ping a combination of the Layer 2 VPN routing instance name, the local site identifier, and the remote site identifier to test the integrity of the Layer 2 VPN connection (specified by the identifiers) between the ingress and egress PE routers.

RELATED DOCUMENTATION

Example: Configure MPLS-Based Layer 2 VPNs | **180**

Pinging a Layer 3 VPN

To ping a Layer 3 VPN, use the following command:

```
ping mpls l3vpn l3vpn-name prefix prefix <count count>
```

You ping a combination of an IPv4 destination prefix and a Layer 3 VPN name on the egress PE router to test the integrity of the VPN connection between the ingress and egress PE routers. The destination prefix corresponds to a prefix in the Layer 3 VPN. However, the ping tests only whether the prefix is present in a PE router's VRF table. It does not test the connection between a PE router and a CE router.

Pinging a Layer 2 Circuit

To ping a Layer 2 circuit, use one of the following commands:

- `ping mpls l2circuit interface interface-name`

  You ping an interface configured for the Layer 2 circuit on the egress PE router.

- `ping mpls l2circuit virtual-circuit neighbor <prefix> <virtual-circuit-id>`

  You ping a combination of the IPv4 prefix and the virtual circuit identifier on the egress PE router to test the integrity of the Layer 2 circuit between the ingress and egress PE routers.

Pinging Customer Edge Device IP Address

IN THIS SECTION

- VPLS or EVPN Use Case | **138**
- H-VPLS Use Case | **139**
- Supported and Unsupported Features for CE-IP Ping | **141**

In a virtual private LAN service (VPLS), hierarchical VPLS (H-VPLS), and Ethernet VPN (EVPN) network, you can test the connectivity to a given customer edge (CE) IP address to get the CE device's MAC address and attachment points (name of the provider edge [PE] device and local interfaces) to the provider network. This is beneficial in Layer 2 VPN technologies, which have a large number of PE devices and for which getting connectivity information about customers is a challenge.

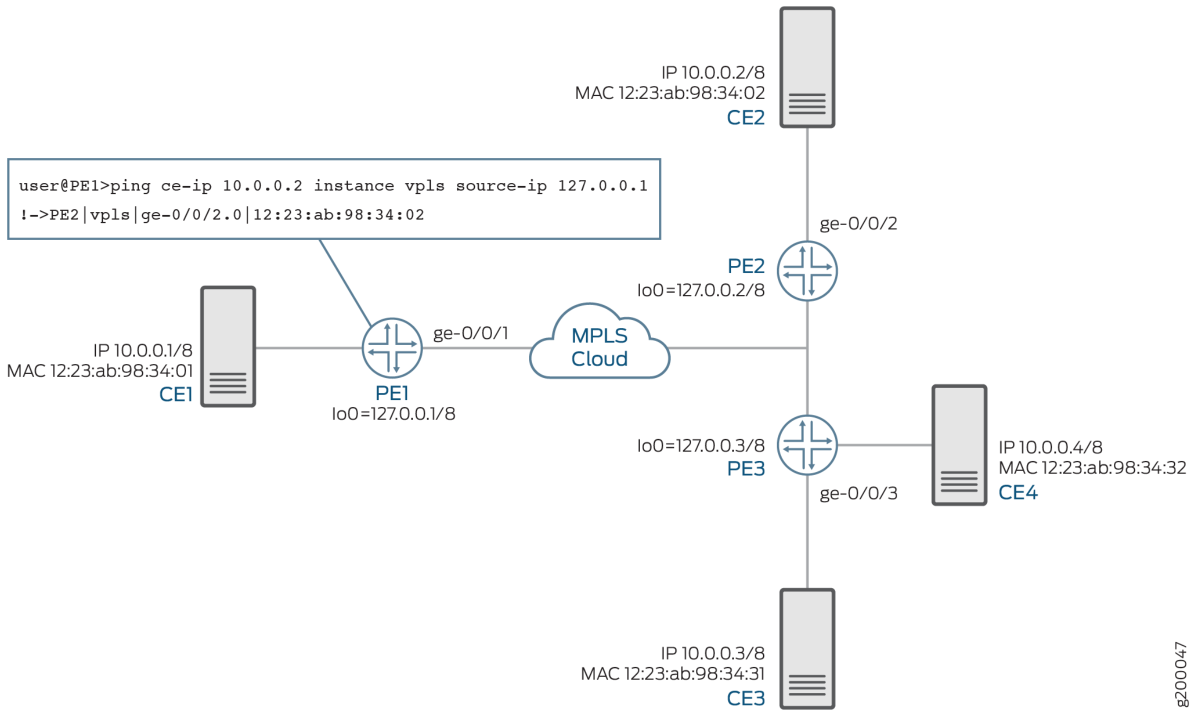
The capability to ping CE IP address has the following use cases and feature support:

VPLS or EVPN Use Case

In older Junos OS releases, the ping utility for VPLS was for destination MAC addresses. Modern Junos OS releases support the CE-IP `ping` utility, which is based on the LSP ping infrastructure defined in RFC 4379. With the CE-IP ping feature, the `ping` utility is enhanced with the capability to ping an IP address for a VPLS and EVPN network. Separate unicast LSP ping echo requests are sent to all neighboring PE devices, and only one PE device responds back with the information about the CE device.

Figure 8 on page 138 illustrates a use case for implementing the CE-IP ping feature in a VPLS or EVPN network. There are three PE devices—Devices PE1, PE2, and PE3—connected to four customer sites—Devices CE1, CE2, CE3, and CE4. In this use case, Device PE1 tests the connectivity to an IP host—10.0.0.2 —to get the MAC address and attachment point of the host in the VPLS or EVPN service provider network for a specific routing instance. This is done using the `ce-ip` command. The command output displays the required information depending on the type of routing instance configured.

Figure 8: CE-IP Ping Feature in a VPLS/EVPN Network



```
user@PE1>ping ce-ip 10.0.0.2 instance vpls source-ip 127.0.0.1
!->PE2|vpls|ge-0/0/2.0|12:23:ab:98:34:02
```

When the `ce-ip` ping command is executed in a VPLS or EVPN network, the packet flow is as follows:

1. **1**—LSP ping echo request

   The `ce-ip` LSP ping echo request packet is sent using the data plane.

   Device PE1 sends an LSP ping echo request to all the neighboring PE devices, Devices PE2 and PE3. The IP address to the target host is carried in the LSP ping echo request using type, length, and value (TLV).

2. **2**—ARP request

   Remote PE devices send host-injected Address Resolution Protocol (ARP) requests on all the CE-facing interfaces for the destination IP address. The ARP request is sent to the host 10.0.0.2 from Device PE2 to Device CE2 and from Device PE3 to Devices CE3 and CE4. The source IP address in the ARP request is set to 0.0.0.0 by default.

3. **3**—ARP response

   Device CE2 responds to the ARP request from Device PE2.

4. **4**—LSP ping echo response

   If an ARP response is received from a CE device, the remote PE device responds to the PE device initiating the ARP request with the MAC address and attachment point encoded as TLV in the LSP ping echo response packet.

   The `ce-ip` LSP ping echo response packet is sent using IP/UDP protocol in the control plane.

   Device PE2 sends an LSP ping response to Device PE1. The other remote PE device, Device PE3, does not respond to the LSP ping because an ARP response is not received from Device CE3.

   The output of the `ce-ip` ping command on Device PE1 displays the information that is received from the LSP ping response.

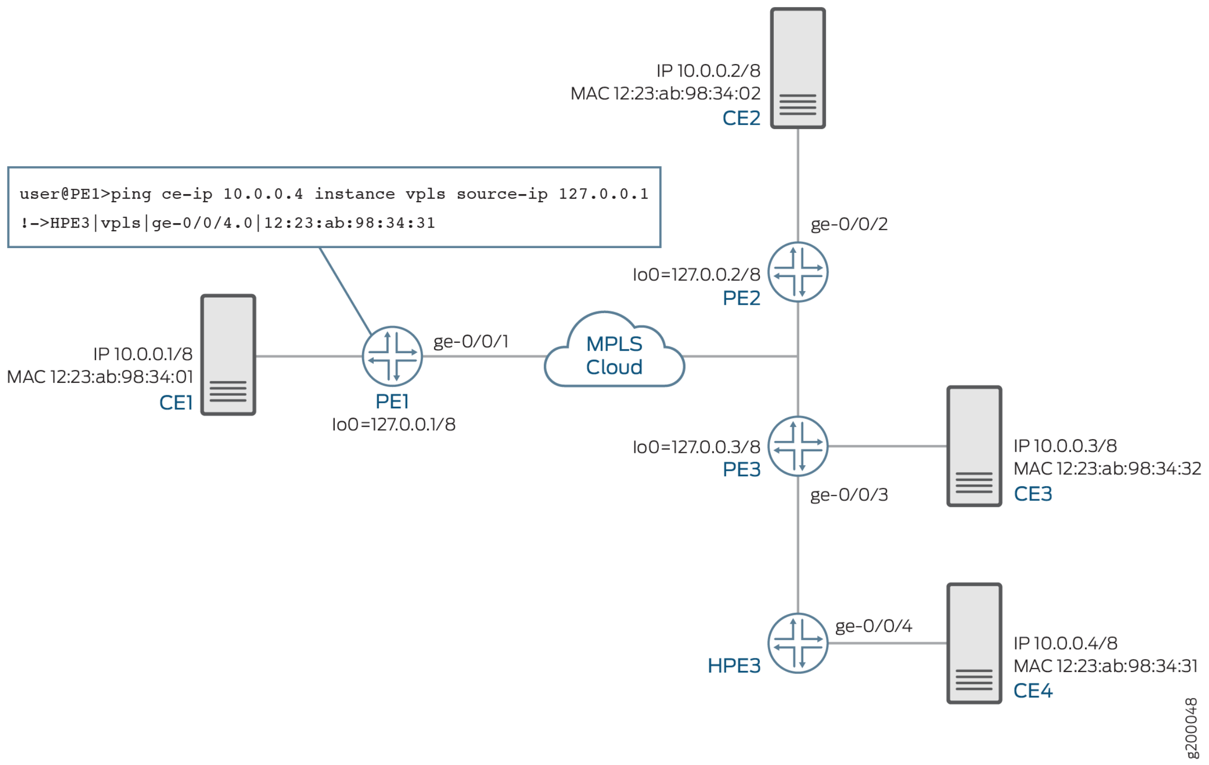
H-VPLS Use Case

In a VPLS or EVPN network, all the PE devices are connected in a mesh topology and therefore the devices are reachable to each other through one hop in terms of virtual circuit label reachability. However, in an H-VPLS network, there are spoke PE devices connected to the VPLS full-mesh network. These spoke PE devices cannot be reached by the remote PE devices through one hop. Because the VPLS ping feature always uses a virtual circuit label TTL value of one, the ping packets are received by the control plane in all the PE devices that are one hop away. The control plane then reinjects the ping packets to the next hop (that is, the spoke PE device) in the H-VPLS network for the ping packet to reach all the PE devices.

Figure 9 on page 140 illustrates a use case for implementing the CE-IP ping feature in an H-VPLS network. There are three PE devices—Devices PE1, PE2, and PE3—connected to four customer sites—Devices CE1, CE2, CE3, and CE4. Device PE3 is connected to an H-VPLS spoke that connects to Device

CE4. In this use case, Device PE1 tests the connectivity to an IP host— 10.0.0.4 —to get the MAC address and attachment point of the host in the H-VPLS service provider network using the `ce-ip` command.

Figure 9: CE-IP Ping Feature in H-VPLS Network



```
user@PE1>ping ce-ip 10.0.0.4 instance vpls source-ip 127.0.0.1
!->HPE3|vpls|ge-0/0/4.0|12:23:ab:98:34:31
```

When the `ce-ip` ping command is executed in an H-VPLS network, the packet flow is as follows:

1. **1**—LSP ping echo request

   The `ce-ip` LSP ping echo request packet is sent using the data plane.

   Device PE1 sends an LSP ping echo request to all the neighboring PE devices, Devices PE2 and PE3. The IP address to the target host is carried in the LSP ping echo request using type, length, and value (TLV).

2. **1A**—LSP re-injected ping request

   Device PE3 re-injects the LSP ping request to the spoke PE device, Device HPE3.

3. **2**—ARP request

The remote PE devices, Devices PE2 and PE3, and the spoke PE device, Device HPE3, send host-injected ARP requests on all the CE-facing interfaces for the destination IP address. The ARP request is sent to the host 10.0.0.4. The source IP address in the ARP request is set to 0.0.0.0 by default.

4. **3**—ARP response

   Device CE4 responds to the ARP request from Device HPE3.

5. **4**—LSP ping echo response

   If an ARP response is received from a CE device, the remote PE device responds to the PE device initiating the ARP request with the MAC address and attachment point encoded as TLV in the LSP ping echo response packet.

   The `ce-ip` LSP ping echo response packet is sent using IP/UDP protocol in the control plane.

   Device HPE3 sends an LSP ping response to Device PE1. The other remote PE devices, Device PE2 and PE3, do not respond to the LSP ping because they do not receive an ARP response from the CE device.

Supported and Unsupported Features for CE-IP Ping

The following features are supported with the CE-IP address ping feature:

- The CE-IP ping feature in a VPLS or H-VPLS network is supported on routing instance type VPLS only
- The CE-IP ping feature in an EVPN network is supported on routing instance type EVPN only.
- Support for VPLS and EVPN hybrid routing instances, where the routing instance type is EVPN and the CE-IP ping support is available on single-homing seamless migration nodes with LDP-VPLS only.

The CE-IP ping feature has the following considerations and limitations:

- If the CE destination IP address that is being pinged is behind the very same PE device where the ping command is issued, the ce-ip ping functionality does not work.
- The LSP ping echo response packet is always sent using the IP/UDP protocol in the control plane. This requires that the PE devices are IP reachable to each other for the feature to work.
- The CE-IP feature does not provide support for the following:
  - Virtual switch routing instance
  - IPv6 addresses
  - Logical systems
  - Integrated routing and bridging (IRB) configured in the EVPN or VPLS routing instance

RELATED DOCUMENTATION

Pinging LSPs

Pinging VPNs, VPLS, and Layer 2 Circuits

Pinging a Layer 2 VPN | **136**

Pinging a VPLS Routing Instance

ping

2

PART

Common Configuration for Layer 2 VPNs and VPLS

- Overview | 144
- Layer 2 VPNs Configuration Overview | 148
- Configuring Layer 2 Interfaces | 210
- Configuring Path Selection for Layer 2 VPNs and VPLS | 216
- Creating Backup Connections with Redundant Pseudowires | 224
- Configuring Class of Service for Layer 2 VPNs | 230
- Monitoring Layer 2 VPNs | 232

CHAPTER 9

Overview

IN THIS CHAPTER

- Understanding Layer 2 VPNs | **144**
- Layer 2 VPN Applications | **145**
- Supported Layer 2 VPN Standards | **146**

Understanding Layer 2 VPNs

As the need to link different Layer 2 services to one another for expanded service offerings grows, Layer 2 Multiprotocol Label Switching (*MPLS*) *VPN* services are increasingly in demand.

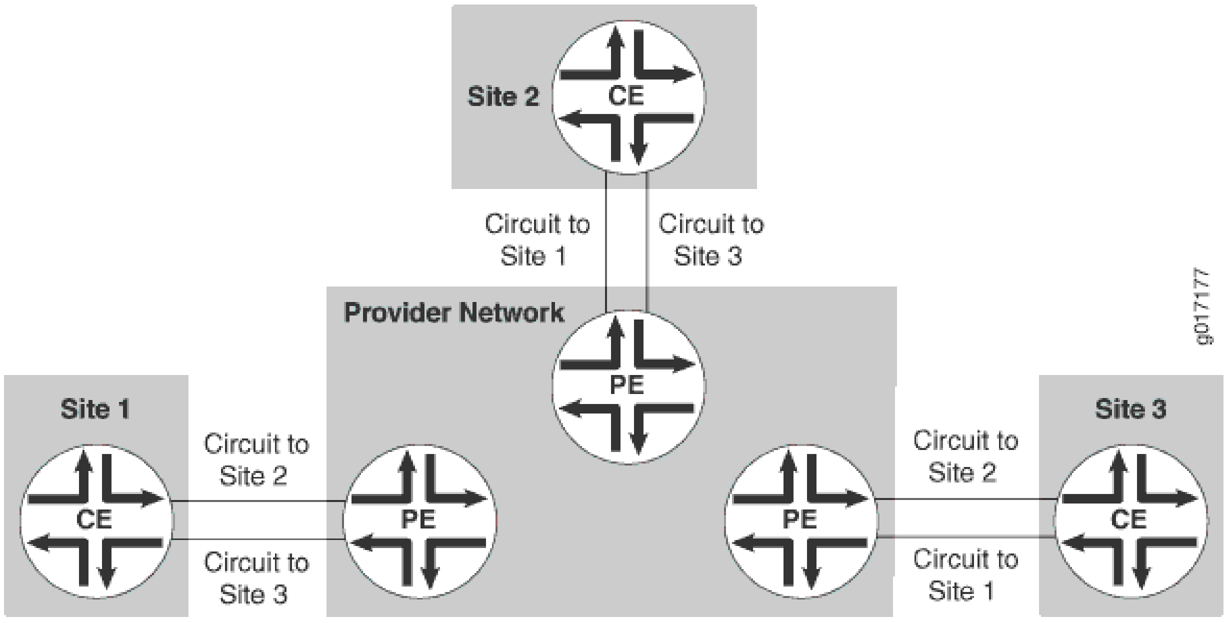
Implementing a Layer 2 VPN on a router is similar to implementing a VPN using other Layer 2 technologies. However, for a Layer 2 VPN on a router, traffic is forwarded to the router in a Layer 2 format. It is carried by MPLS over the service provider's network, and then converted back to Layer 2 format at the receiving site. You can configure different Layer 2 formats at the sending and receiving sites. The service provisioned with Layer 2 VPNs is also known as *Virtual Private Wire Service* (*VPWS*).

On a Layer 2 VPN, routing occurs on the customer's routers, typically on the customer edge (CE) router. The CE router connected to a service provider on a Layer 2 VPN must select the appropriate circuit on which to send traffic. The provider edge (PE) router receiving the traffic sends it across the service provider's network to the PE router connected to the receiving site. The PE routers do not need to store or process the customer's routes; they only need to be configured to send data to the appropriate tunnel.

For a Layer 2 VPN, customers need to configure their own routers to carry all Layer 3 traffic. The service provider needs to know only how much traffic the Layer 2 VPN will need to carry. The service provider's routers carry traffic between the customer's sites using Layer 2 VPN interfaces. The VPN topology is determined by policies configured on the PE routers.

Customers need to know only which VPN interfaces connect to which of their own sites. Figure 10 on page 145 illustrates a Layer 2 VPN in which each site has a VPN interface linked to each of the other customer sites.

Figure 10: Layer 2 VPN Connecting CE Routers



Implementing a Layer 2 MPLS VPN includes the following benefits:

- Service providers do not have to invest in separate Layer 2 equipment to provide Layer 2 VPN service. A Layer 2 MPLS VPN allows you to provide Layer 2 VPN service over an existing IP and MPLS backbone.
- You can configure the PE router to run any Layer 3 protocol in addition to the Layer 2 protocols.
- Customers who prefer to maintain control over most of the administration of their own networks might want Layer 2 VPN connections with their service provider instead of a Layer 3 VPN.
- Because Layer 2 VPNs use *BGP* as the signaling protocol, they have a simpler design and require less overhead than traditional VPNs over Layer 2 circuits. BGP signaling also enables autodiscovery of Layer 2 VPN peers. Layer 2 VPNs are similar to BGP or MPLS VPNs and *VPLS* in many respects; all three types of services employ BGP for signaling.

Layer 2 VPN Applications

Implementing a Layer 2 VPN includes the following benefits:

- Terminating a Layer 2 VPN into a Layer 2 VPN using the interworking (iw0) software interface eliminates the limitation of bandwidth on the tunnel interfaces used for these configuration scenarios. Instead of using a physical Tunnel PIC for looping the packet received from the Layer 2 VPN to another Layer 2 VPN, Junos OS is used to link both the Layer 2 VPN routes.

- Layer 2 VPNs enable the sharing of a provider's core network infrastructure between IP and Layer 2 VPN services, reducing the cost of providing those services. A Layer 2 MPLS VPN allows you to provide Layer 2 VPN service over an existing IP and MPLS backbone.
- From a service provider's point of view, a Layer 2 MPLS VPN allows the use of a single Layer 3 VPN (such as RFC 2547bis), MPLS traffic engineering, and Differentiated Services (DiffServ).
- Service providers do not have to invest in separate Layer 2 equipment to provide Layer 2 VPN service. You can configure the PE router to run any Layer 3 protocol in addition to the Layer 2 protocols. Customers who prefer to maintain control over most of the administration of their own networks might want Layer 2 VPN connections with their service provider instead of a Layer 3 VPN.

RELATED DOCUMENTATION

Understanding Layer 2 VPNs | **144**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 Circuit to a Layer 2 VPN | **1333**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 VPN to a Layer 2 VPN | **1268**

Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN | **1335**

Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN | **1270**

Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN

Supported Layer 2 VPN Standards

Junos OS substantially supports the following standards and Internet drafts, which define standards for Layer 2 virtual private networks (VPNs).

- RFC 7348, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*
- Internet draft draft-kompella-l2vpn-vpls-multihoming, *Multi-homing in BGP-based Virtual Private LAN Service*
- Internet draft draft-kompella-ppvpn-l2vpn-03.txt, *Layer 2 VPNs Over Tunnels*

RELATED DOCUMENTATION

Supported Carrier-of-Carriers and Interprovider VPN Standards

Supported VPWS Standards | **495**

Supported Layer 3 VPN Standards

Supported Multicast VPN Standards

Supported VPLS Standards | **607**

Accessing Standards Documents on the Internet

CHAPTER 10

Layer 2 VPNs Configuration Overview

IN THIS CHAPTER

- Introduction to Configuring Layer 2 VPNs | **148**
- Configuring the Local Site on PE Routers in Layer 2 VPNs | **150**
- Layer 2 VPN Configuration Example | **157**
- Example: Configure MPLS-Based Layer 2 VPNs | **180**
- Transmitting Nonstandard BPDUs in Layer 2 VPNs and VPLS | **209**

Introduction to Configuring Layer 2 VPNs

To configure Layer 2 virtual private network (VPN) functionality, you must enable Layer 2 VPN support on the provider edge (PE) router. You must also configure PE routers to distribute routing information to the other PE routers in the VPN and configure the circuits between the PE routers and the customer edge (CE) routers.

Each Layer 2 VPN is configured under a routing instance of type `l2vpn`. An `l2vpn` routing instance can transparently carry Layer 3 traffic across the service provider's network. As with other routing instances, all logical interfaces belonging to a Layer 2 VPN routing instance are listed under that instance.

The configuration of the CE routers is not relevant to the service provider. The CE routers need to provide only appropriate Layer 2 circuits (with appropriate circuit identifiers, such as virtual path identifier/virtual channel identifier [VPI/VCI], or virtual LAN [VLAN] ID) to send traffic to the PE router.

To configure Layer 2 VPNs, include the following statements:

```
description text;
instance-type l2vpn;
interface interface-name;
route-distinguisher (as-number:id| ip-address:id);
vrf-export [ policy-names ];
vrf-import [ policy-names ];
vrf-target {
        community;
        import community-name;
        export community-name;
    }
    protocols {
        l2vpn {
            (control-word | no-control-word);
            encapsulation-type type;
            site site-name {
                interface interface-name {
                    description text;
                    remote-site-id remote-site-id;
                }
                site-identifier identifier;
                site-preference preference-value {
                    backup;
                    primary;
                }
            }
            traceoptions {
                file filename <files number> <size size> <world-readable | no-world-readable>;
                flag flag <flag-modifier> <disable>;
            }
        }
    }
```

You can include these statements at the following hierarchy levels:

- `[edit routing-instances routing-instance-name]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name]`

For Layer 2 VPNs, only some of the statements in the `[edit routing-instances]` hierarchy are valid. For the full hierarchy, see Junos OS Routing Protocols Library.

In addition to these statements, you must configure MPLS label-switched paths (LSPs) between the PE routers, IBGP sessions between the PE routers, and an interior gateway protocol (IGP) on the PE and provider (P) routers. You must also configure the statements that are required for all types of VPN configuration.

By default, Layer 2 VPNs are disabled.

Many of the configuration procedures for Layer 2 VPNs are identical to the procedures for Layer 3 VPNs and virtual private LAN service (VPLS).

Configuring the Local Site on PE Routers in Layer 2 VPNs

IN THIS SECTION

- Configuring a Layer 2 VPN Routing Instance | **150**
- Configuring the Site | **151**
- Configuring the Remote Site ID | **152**
- Configuring the Encapsulation Type | **154**
- Configuring a Site Preference and Layer 2 VPN Multihoming | **155**
- Tracing Layer 2 VPN Traffic and Operations | **156**

For each local site, the PE router advertises a set of VPN labels to the other PE routers servicing the Layer 2 VPN. The VPN labels constitute a single block of contiguous labels; however, to allow for reprovisioning, more than one such block can be advertised. Each label block consists of a label base, a range (the size of the block), and a remote site ID that identifies the sequence of remote sites that connect to the local site using this label block (the remote site ID is the first site identifier in the sequence). The encapsulation type is also advertised along with the label block.

The following sections explain how to configure the connections to the local site on the PE router.

NOTE: Not all subtasks are supported on all platforms; check the CLI on your device.

Configuring a Layer 2 VPN Routing Instance

To configure a Layer 2 VPN on your network, configure a Layer 2 VPN routing instance on the PE router by including the `l2vpn` statement:

NOTE: On the EX9200 switches, replace *encapsulation-type* with the *encapsulation* statement.

```
l2vpn {
    (control-word | no-control-word);
    encapsulation-type  type;
    traceoptions {
        file filename <files number> <size size> <world-readable | no-world-readable>;
```

```
            flag flag <flag-modifier> <disable>;
        }
        site site-name {
            site-identifier identifier;
            site-preference preference-value {
                backup;
                primary;
            }
            interface interface-name {
                description text;
                remote-site-id remote-site-id;
            }
        }
    }
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols]`

NOTE: You cannot configure a routing protocol (OSPF, RIP, IS-IS or BGP) inside a Layer 2 VPN routing instance (`instance-type l2vpn`). The Junos CLI disallows this configuration.

Instructions for how to configure the remaining statements are included in the sections that follow.

Configuring the Site

All the Layer 2 circuits provisioned for a local site are listed as the set of logical interfaces (specified by including the `interface` statement) within the `site` statement.

On each PE router, you must configure each site that has a circuit to the PE router. To do this, include the `site` statement:

```
site site-name {
    site-identifier identifier;
    site-preference preference-value {
        backup;
        primary;
    }
    interface interface-name {
        description text;
```

```
            remote-site-id remote-site-ID;
        }
    }
```

You can include this statement at 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]`

You must configure the following for each site:

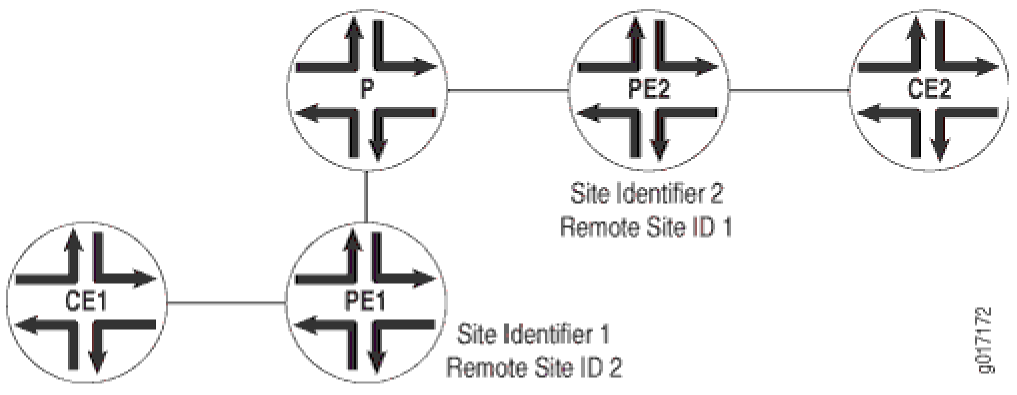
- *site-name*—Name of the site.
- `site-identifier` *identifier*—Unsigned 16-bit number greater than zero that uniquely identifies the local Layer 2 VPN site. The site identifier corresponds to the remote site ID configured on another site within the same VPN.
- `interface` *interface-name*—The name of the interface and, optionally, a remote site ID for remote site connections. See "Configuring the Remote Site ID" on page 152.

Configuring the Remote Site ID

The remote site ID allows you to configure a sparse Layer 2 VPN topology. A sparse topology means that each site does not have to connect to all the other sites in the VPN; thus it is unnecessary to allocate circuits for all the remote sites. Remote site IDs are particularly important if you configure a topology more complicated than full-mesh, such as a hub-and-spoke topology.

The remote site ID (configured with the `remote-site-id` statement) corresponds to the site ID (configured with the `site-identifier` statement) configured at a separate site. Figure 11 on page 152 illustrates the relationship between the site identifier and the remote site ID.

Figure 11: Relationship Between the Site Identifier and the Remote Site ID



As illustrated by the figure, the configuration for Router PE1 connected to Router CE1 is as follows:

```
site-identifier 1;
interface so-0/0/0 {
    remote-site-id 2;
}
```

The configuration for Router PE2 connected to Router CE2 is as follows:

```
site-identifier 2;
interface so-0/0/1 {
    remote-site-id 1;
}
```

The remote site ID (2) on Router PE1 corresponds to the site identifier (2) on Router PE2. On Router PE2, the remote site ID (1) corresponds to the site identifier (1) on Router PE1.

To configure the remote site ID, include the `remote-site-id` statement:

```
remote-site-id remote-site-id;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols l2vpn site` *`site-name`* `interface` *`interface-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols l2vpn site` *`site-name`* `interface` *`interface-name`*`]`

If you do not explicitly include the `remote-site-id` statement for the interface configured at the `[edit routing-instances` *`routing-instance-name`* `protocols l2vpn site` *`site-name`*`]` hierarchy level, a remote site ID is assigned to that interface.

The remote site ID for an interface is automatically set to 1 higher than the remote site ID for the previous interface. The order of the interfaces is based on their `site-identifier` statements. For example, if the first interface in the list does not have a remote site ID, its ID is set to 1. The second interface in the list has its remote site ID set to 2, and the third has its remote site ID set to 3. The remote site IDs of any interfaces that follow are incremented in the same manner if you do not explicitly configure them.

Configuring the Encapsulation Type

The encapsulation type you configure at each Layer 2 VPN site varies depending on which Layer 2 protocol you choose to configure. If you configure `ethernet-vlan` as the encapsulation type, you need to use the same protocol at each Layer 2 VPN site.

You do not need to use the same protocol at each Layer 2 VPN site if you configure any of the following encapsulation types:

- `atm-aal5`—Asynchronous Transfer Mode (ATM) Adaptation Layer (AAL5)
- `atm-cell`—ATM cell relay
- `atm-cell-port-mode`—ATM cell relay port promiscuous mode
- `atm-cell-vc-mode`—ATM virtual circuit (VC) cell relay nonpromiscuous mode
- `atm-cell-vp-mode`—ATM virtual path (VP) cell relay promiscuous mode
- `cisco-hdlc`—Cisco Systems-compatible High-Level Data Link Control (HDLC)
- `ethernet`—Ethernet
- `ethernet-vlan`—Ethernet virtual LAN (VLAN)
- `frame-relay`—Frame Relay
- `frame-relay-port-mode`—Frame Relay port mode
- `interworking`—Layer 2.5 interworking VPN
- `ppp`—Point-to-Point Protocol (PPP)

If you configure different protocols at your Layer 2 VPN sites, you need to configure a translational cross-connect (TCC) encapsulation type. For more information, see "Configuring TCC Encapsulation for Layer 2 VPNs and Layer 2 Circuits" on page 211.

To configure the Layer 2 protocol accepted by the PE router, specify the encapsulation type by including the `encapsulation-type` statement:

```
encapsulation-type type;
```

For EX9200 switches, specify the encapsulation type by including the `encapsulation` statement:

```
encapsulation type;
```

You can include this statement at 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]`

Configuring a Site Preference and Layer 2 VPN Multihoming

You can specify the preference value advertised for a particular Layer 2 VPN site. The site preference value is encoded in the BGP local preference attribute. When a PE router receives multiple advertisements with the same CE device identifier, the advertisement with the highest local preference value is preferred.

You can also use the `site-preference` statement to enable multihoming for Layer 2 VPNs. Multihoming allows you to connect a CE device to multiple PE routers. In the event that a connection to the primary PE router fails, traffic can be automatically switched to the backup PE router.

To configure a site preference for a Layer 2 VPN, include the `site-preference` statement:

```
site-preference preference-value {
    backup;
    primary;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances *routing-instance-name* protocols l2vpn site *site-name*]`
- `[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols l2vpn site *site-name*]`

You can also specify either the `backup` option or the `primary` option for the `site-preference` statement. The backup option specifies the preference value as 1, the lowest possible value, ensuring that the Layer 2 VPN site is the least likely to be selected. The primary option specifies the preference value as 65,535, the highest possible value, ensuring that the Layer 2 VPN site is the most likely to be selected.

For Layer 2 VPN multihoming configurations, specifying the `primary` option for a Layer 2 VPN site designates the connection from the PE router to the CE device as the preferred connection if the CE device is also connected to another PE router. Specifying the `backup` option for a Layer 2 VPN site designates the connection from the PE router to the CE device as the secondary connection if the CE device is also connected to another PE router.

Tracing Layer 2 VPN Traffic and Operations

To trace Layer 2 VPN protocol traffic, specify options for the `traceoptions` statement in the Layer 2 VPN configuration:

```
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
```

You can include this statement at 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]`

The following trace flags display the operations associated with Layer 2 VPNs:

- `all`—All Layer 2 VPN tracing options.
- `connections`—Layer 2 connections (events and state changes).
- `error`—Error conditions.
- `general`—General events.
- `nlri`—Layer 2 advertisements received or sent by means of the BGP.
- `normal`—Normal events.
- `policy`—Policy processing.
- `route`—Routing information.
- `state`—State transitions.
- `task`—Routing protocol task processing.
- `timer`—Routing protocol timer processing.
- `topology`—Layer 2 VPN topology changes caused by reconfiguration or advertisements received from other PE routers using BGP.

Disabling Normal TTL Decrementing for VPNs

To diagnose networking problems related to VPNs, it can be useful to disable normal time-to-live (TTL) decrementing. In Junos, you can do this with the `no-propagate-ttl` and `no-decrement-ttl` statements. However, when you are tracing VPN traffic, only the `no-propagate-ttl` statement is effective.

For the `no-propagate-ttl` statement to have an effect on VPN behavior, you need to clear the PE-router-to-PE-router BGP session, or disable and then enable the VPN routing instance.

For more information about the `no-propagate-ttl` and `no-decrement-ttl` statements, see the MPLS Applications User Guide..

Layer 2 VPN Configuration Example

IN THIS SECTION

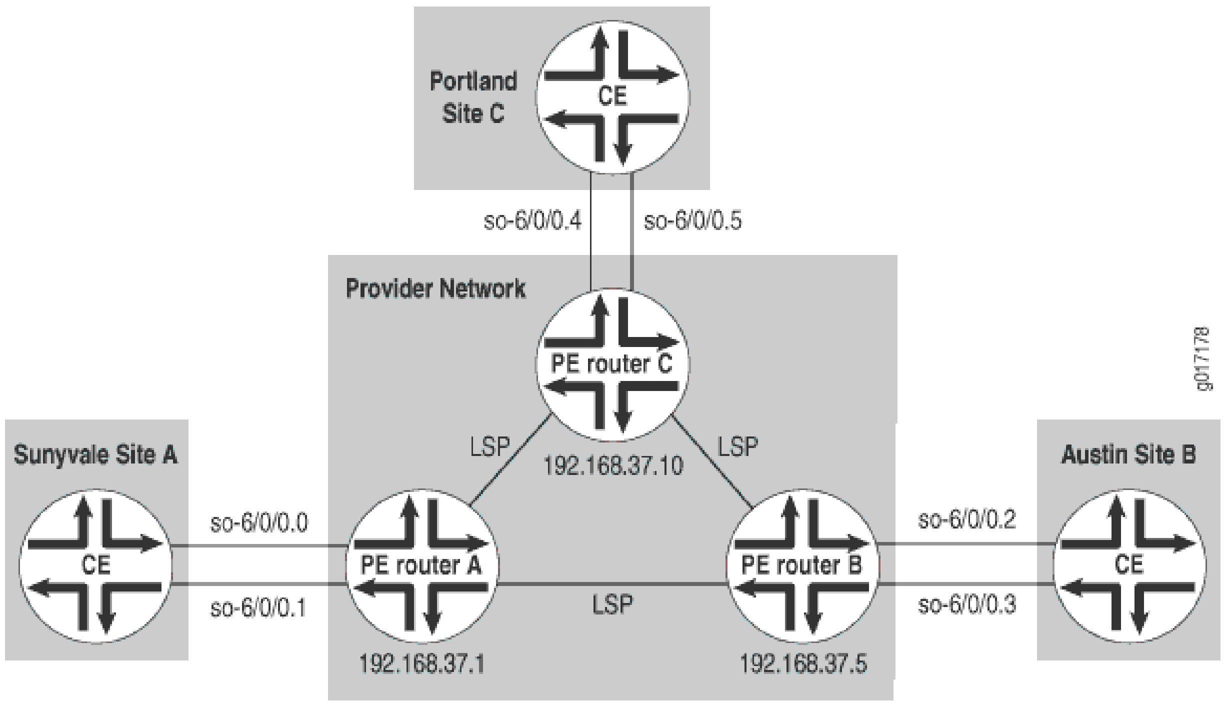
- Simple Full-Mesh Layer 2 VPN Overview | 157
- Enabling an IGP on the PE Routers | 158
- Configuring MPLS LSP Tunnels Between the PE Routers | 158
- Configuring IBGP on the PE Routers | 160
- Configuring Routing Instances for Layer 2 VPNs on the PE Routers | 162
- Configuring CCC Encapsulation on the Interfaces | 165
- Configuring VPN Policy on the PE Routers | 166
- Layer 2 VPN Configuration Summarized by Router | 170

The following sections explain how to configure Layer 2 VPN functionality on the provider edge (PE) routers connected to each site:

Simple Full-Mesh Layer 2 VPN Overview

In the sections that follow, you configure a simple full-mesh Layer 2 VPN spanning three sites: Sunnyvale, Austin, and Portland. Each site connects to a PE router. The customer edge (CE) routers at each site use Frame Relay to carry Layer 2 traffic to the PE routers. Since this example uses a full-mesh topology between all three sites, each site requires two logical interfaces (one for each of the other CE routers), although only one physical link is needed to connect each PE router to each CE router. Figure 12 on page 158 illustrates the topology of this Layer 2 VPN.

Figure 12: Example of a Simple Full-Mesh Layer 2 VPN Topology



Enabling an IGP on the PE Routers

To allow the PE routers to exchange routing information among themselves, you must configure an interior gateway protocol (IGP) or static routes on these routers. You configure the IGP on the primary instance of the routing protocol process (rpd) (that is, at the `[edit protocols]` hierarchy level), not within the Layer 2 VPN routing instance (that is, not at the `[edit routing-instances]` hierarchy level). Turn on traffic engineering on the IGP.

You configure the IGP in the standard way. This example does not include this portion of the configuration.

Configuring MPLS LSP Tunnels Between the PE Routers

In this configuration example, RSVP is used for MPLS signaling. Therefore, in addition to configuring RSVP, you must create an MPLS label-switched path (LSP) to tunnel the VPN traffic.

On Router A, enable RSVP and configure one end of the MPLS LSP tunnel to Router B. When configuring the MPLS LSP, include all interfaces using the `interface all` statement.

```
[edit]
protocols {
    rsvp {
        interface all;
    }
    mpls {
        interface all;
        label-switched-path RouterA-to-RouterB {
            to 192.168.37.5;
            primary Path-to-RouterB;
        }
        label-switched-path RouterA-to-RouterC {
            to 192.168.37.10;
            primary Path-to-RouterC;
        }
    }
}
```

On Router B, enable RSVP and configure the other end of the MPLS LSP tunnel. Again, configure the interfaces by using the `interface all` statement.

```
[edit]
protocols {
    rsvp {
        interface all;
    }
    mpls {
        interface all;
        label-switched-path RouterB-to-RouterA {
            to 192.168.37.1;
            primary Path-to-RouterA;
        }
        label-switched-path RouterB-to-RouterC {
            to 192.168.37.10;
            primary Path-to-RouterC;
        }
    }
}
```

On Router C, enable RSVP and configure the other end of the MPLS LSP tunnel. Again, configure all interfaces using the `interface all` statement.

```
[edit]
protocols {
    rsvp {
        interface all;
    }
    mpls {
        interface all;
        label-switched-path RouterC-to-RouterA {
            to 192.168.37.1;
            primary Path-to-RouterA;
        }
        label-switched-path RouterC-to-RouterB {
            to 192.168.37.5;
            primary Path-to-RouterB;
        }
    }
}
```

Configuring IBGP on the PE Routers

On the PE routers, configure an IBGP session with the following parameters:

- Layer 2 VPN—To indicate that the IBGP session is for a Layer 2 VPN, include the `family l2vpn` statement.
- Local address—The IP address in the `local-address` statement is the same as the address configured in the `to` statement at the `[edit protocols mpls label-switched-path` *`lsp-path-name`*`]` hierarchy level on the remote PE router. The IBGP session for Layer 2 VPNs runs through this address.
- Neighbor address—Include the `neighbor` statement, specifying the IP address of the neighboring PE router.

On Router A, configure IBGP:

```
[edit]
protocols {
    bgp {
        import match-all;
        export match-all;
        group pe-pe {
            type internal;
            neighbor 192.168.37.5 {
                local-address 192.168.37.1;
                family l2vpn {
                    signaling;
                }
            }
            neighbor 192.168.37.10 {
                local-address 192.168.37.1;
                family l2vpn {
                    signaling;
                }
            }
        }
    }
}
```

On Router B, configure IBGP:

```
[edit]
protocols {
    bgp {
        local-address 192.168.37.5;
        import match-all;
        export match-all;
        group pe-pe {
            type internal;
            neighbor 192.168.37.1 {
                local-address 192.168.37.5;
                family l2vpn {
                    signaling;
                }
            }
            neighbor 192.168.37.10 {
                local-address 192.168.37.5;
                family l2vpn {
                    signaling;
                }
            }
        }
    }
}
```

On Router C, configure IBGP:

```
[edit]
protocols {
    bgp {
        local-address 192.168.37.10;
        import match-all;
        export match-all;
        group pe-pe {
            type internal;
            neighbor 192.168.37.1 {
                local-address 192.168.37.10;
                family l2vpn {
                    signaling;
                }
            }
            neighbor 192.168.37.5 {
                local-address 192.168.37.10;
                family l2vpn {
                    signaling;
                }
            }
        }
    }
}
```

Configuring Routing Instances for Layer 2 VPNs on the PE Routers

The three PE routers service the Layer 2 VPN, so you need to configure a routing instance on each router. For the VPN, you must define the following in each routing instance:

- Route distinguisher, which must be unique for each routing instance on the PE router. It is used to distinguish the addresses in one VPN from those in another VPN.
- Instance type of `l2vpn`, which configures the router to run a Layer 2 VPN.
- Interfaces connected to the CE routers.

- Virtual routing and forwarding (*VRF*) import and export policies, which must be the same on each PE router that services the same VPN and are used to control the network topology. Unless the import policy contains only a `then reject` statement, it must include a reference to a community. Otherwise, when you attempt to commit the configuration, the commit operation fails.

On Router A, configure the following routing instance for the Layer 2 VPN:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.0;
        interface so-6/0/0.1;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
        protocols {
            l2vpn {
                encapsulation-type frame-relay;
                site Sunnyvale {
                    site-identifier 1;
                    interface so-6/0/0.0 {
                        remote-site-id 2;
                    }
                    interface so-6/0/0.1 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

On Router B, configure the following routing instance for the Layer 2 VPN:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.2;
        interface so-6/0/0.3;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
        protocols {
            l2vpn {
                encapsulation-type frame-relay;
                site Austin {
                    site-identifier 2;
                    interface so-6/0/0.2 {
                        remote-site-id 1;
                    }
                    interface so-6/0/0.3 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

On Router C, configure the following routing instance for the Layer 2 VPN:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.4;
        interface so-6/0/0.5;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
        protocols {
            l2vpn {
                encapsulation-type frame-relay;
                site Portland {
                    site-identifier 3;
                    interface so-6/0/0.4 {
                        remote-site-id 1;
                    }
                    interface so-6/0/0.5 {
                        remote-site-id 2;
                    }
                }
            }
        }
    }
}
```

Configuring CCC Encapsulation on the Interfaces

You need to specify a circuit cross-connect (CCC) encapsulation type for each PE-router-to-CE-router interface running in the Layer 2 VPN. This encapsulation type should match the encapsulation type configured under the routing instance.

Configure the following CCC encapsulation types for the interfaces on Router A:

```
[edit]
interfaces so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 0 {
        encapsulation frame-relay-ccc;
    }
}
interfaces so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 1 {
        encapsulation frame-relay-ccc;
    }
}
```

Configure the following CCC encapsulation types for the interfaces on Router B:

```
[edit]
interfaces so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 2 {
        encapsulation frame-relay-ccc;
    }
}
interfaces so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 3 {
        encapsulation frame-relay-ccc;
    }
}
```

Configure the following CCC encapsulation types for the interfaces on Router C:

```
[edit]
interface so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 4 {
        encapsulation frame-relay-ccc;
    }
}
interface so-6/0/0 {
    encapsulation frame-relay-ccc;
    unit 5 {
        encapsulation frame-relay-ccc;
    }
}
```

Configuring VPN Policy on the PE Routers

You must configure VPN import and export policies on each of the PE routers so that they install the appropriate routes in their VRF tables, which the routers use to forward packets within the VPN.

NOTE: Use the `community add` *`community-name`* statement at the `[edit policy-options policy-statement` *`policy-statement-name`* `term` *`term-name`* `then]` hierarchy level to facilitate Layer 2 VPN VRF export policies.

On Router A, configure the following VPN import and export policies:

```
[edit]
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

On Router B, configure the following VPN import and export policies:

```
[edit]
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

On Router C, configure the following VPN import and export policies:

```
[edit]
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

To apply the VPN policies on the routers, include the `vrf-export` and `vrf-import` statements when you configure the routing instance. The VRF import and export policies handle the route distribution across the IBGP session running between the PE routers.

To apply the VPN policies on Router A, include the following statements:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
    }
}
```

To apply the VPN policies on Router B, include the following statements:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
    }
}
```

To apply the VPN policies on Router C, include the following statements:

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
    }
}
```

Layer 2 VPN Configuration Summarized by Router

For a summary of the configuration on each router in the examples in this chapter, see the following sections:

Summary for Router A (PE Router for Sunnyvale)

Routing Instance for Layer 2 VPN

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.0;
        interface so-6/0/0.1;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
        protocols {
            l2vpn {
                encapsulation-type frame-relay;
                site Sunnyvale {
                    site-identifier 1;
                    interface so-6/0/0.0 {
                        remote-site-id 2;
                    }
                    interface so-6/0/0.1 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

Configure CCC Encapsulation Types for Interfaces

```
interfaces {
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 0 {
            encapsulation frame-relay-ccc;
        }
    }
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 1 {
            encapsulation frame-relay-ccc;
        }
    }
}
```

Primary Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
    interface all;
}
```

Configure MPLS LSPs

```
mpls {
    label-switched-path RouterA-to-RouterB {
        to 192.168.37.5;
        primary Path-to-RouterB {
            cspf;
        }
    }
    label-switched-path RouterA-to-RouterC {
        to 192.168.37.10;
        primary Path-to-RouterC {
            cspf;
        }
    }
    interface all;
}
```

Configure IBGP

```
bgp {
    import match-all;
    export match-all;
    group pe-pe {
        type internal;
        neighbor 192.168.37.5 {
            local-address 192.168.37.1;
            family l2vpn {
                signaling;
            }
        }
        neighbor 192.168.37.10 {
            local-address 192.168.37.1;
            family l2vpn {
                signaling;
            }
        }
    }
}
```

Configure VPN Policy

```
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

Summary for Router B (PE Router for Austin)

Routing Instance for VPN

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.2;
        interface so-6/0/0.3;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
    }
}
```

Configure Layer 2 VPN

```
protocols {
    l2vpn {
        encapsulation-type frame-relay;
        site Austin {
            site-identifier 2;
            interface so-6/0/0.2 {
                remote-site-id 1;
            }
            interface so-6/0/0.3 {
                remote-site-id 3;
            }
        }
    }
}
```

Configure CCC Encapsulation Types for Interfaces

```
[edit]
interfaces {
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 2 {
            encapsulation frame-relay-ccc;
        }
    }
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 3 {
            encapsulation frame-relay-ccc;
        }
    }
}
```

Primary Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
    interface all;
}
```

Configure MPLS LSPs

```
mpls {
    label-switched-path RouterB-to-RouterA {
        to 192.168.37.1;
        primary Path-to-RouterA {
            cspf;
        }
    }
    label-switched-path RouterB-to-RouterC {
        to 192.168.37.10;
        primary Path-to-RouterC {
            cspf;
        }
    }
    interface all;
}
```

Configure IBGP

```
bgp {
    local-address 192.168.37.5;
    import match-all;
    export match-all;
    group pe-pe {
        type internal;
        neighbor 192.168.37.1 {
            local-address 192.168.37.5;
            family l2vpn {
                signaling;
            }
        }
        neighbor 192.168.37.10 {
            local-address 192.168.37.5;
            family l2vpn {
                signaling;
            }
        }
    }
}
```

Configure VPN Policy

```
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

Summary for Router C (PE Router for Portland)

Routing Instance for VPN

```
[edit]
routing-instances {
    VPN-Sunnyvale-Portland-Austin {
        instance-type l2vpn;
        interface so-6/0/0.3;
        interface so-6/0/0.4;
        route-distinguisher 100:1;
        vrf-import vpn-SPA-import;
        vrf-export vpn-SPA-export;
    }
}
```

Configure Layer 2 VPN

```
protocols {
    l2vpn {
        encapsulation-type frame-relay;
        site Portland {
            site-identifier 3;
            interface so-6/0/0.4 {
                remote-site-id 1;
            }
            interface so-6/0/0.5 {
                remote-site-id 2;
            }
        }
    }
}
```

Configure CCC Encapsulation Types for Interfaces

```
[edit]
interfaces {
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 4 {
            encapsulation frame-relay-ccc;
        }
    }
    interface so-6/0/0 {
        encapsulation frame-relay-ccc;
        unit 5 {
            encapsulation frame-relay-ccc;
        }
    }
}
```

Primary Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
    interface all;
}
```

Configure MPLS LSPs

```
mpls {
    label-switched-path RouterC-to-RouterA {
        to 192.168.37.1;
        primary Path-to-RouterA {
            cspf;
        }
    }
    label-switched-path RouterC-to-RouterB {
        to 192.168.37.5;
        primary Path-to-RouterB {
            cspf;
        }
    }
    interface all;
}
```

Configure IBGP

```
bgp {
    local-address 192.168.37.10;
    import match-all;
    export match-all;
    group pe-pe {
        type internal;
        neighbor 192.168.37.1 {
            local-address 192.168.37.10;
            family l2vpn {
                signaling;
            }
        }
        neighbor 192.168.37.5 {
            local-address 192.168.37.10;
            family l2vpn {
                signaling;
            }
        }
    }
}
```

Configure VPN Policy

```
policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:69:100;
}
```

Example: Configure MPLS-Based Layer 2 VPNs

IN THIS SECTION

- Requirements | 181
- Overview and Topology | 182
- Quick Configurations | 183
- Configure the Local PE (PE1) Device for a MPLS-Based Layer 2 VPN | 186
- Configure the Remote PE (PE2) Device for a MPLS-Based Layer 2 VPN | 194
- Verification | 200

This example shows how to configure and validate an MPLS-based Layer 2 VPN on routers or switches running Junos OS.

NOTE: Our content testing team has validated and updated this example.

You can deploy an MPLS-based Layer 2 virtual private network using routers and switches running Junos OS to interconnect customer sites with Layer 2 connectivity. Layer 2 VPNs give customers complete control over their choice of transport and routing protocols.

MPLS-based VPNs require baseline MPLS functionality in the provider network. Once basic MPLS is operational, you are able to configure VPNs that use Label-switched paths (LSPs) for transport over the provider's core.

The addition of VPN services does not affect the basic MPLS switching operations in the provider network. In fact, the provider (P) devices require only a baseline MPLS configuration because they are not VPN aware. VPN state is maintained only on the PE devices. This is a key reason why MPLS-based VPNs are so scalable.

Requirements

This example uses the following hardware and software components:

- Junos OS Release 15.1 or later
  - Revalidated on Junos OS Release 20.1R1
- Two Provider edge (PE) devices
- One provider (P) device
- Two customer edge (CE) devices

The example focuses on how to add Layer 2 VPN to a pre-existing MPLS baseline. A basic MPLS configuration is provided in case your network does not already have MPLS deployed.

To support MPLS-based VPNs the underlying MPLS baseline must provide the following functionality:

- Core-facing and loopback interfaces operational with MPLS family support
- An interior gateway protocol such as OSPF or IS-IS to provide reachability between the loopback addresses of the provider (P and PE) devices
- An MPLS signalling protocol such as LDP or RSVP to signal LSPs
- LSPs established between PE device loopback addresses

LSPs are needed between each pair of PE devices that participate in a given VPN. Its a good idea to build LSPs between all PE devices to accommodate future VPN growth. You configure LSPs at the `[edit protocols mpls]` hierarchy level. Unlike an MPLS configuration for circuit cross-connect (CCC) , you do not need to manually associate the LSP with the PE device's customer-facing (edge) interface. Instead, Layer 2 VPNs use BGP signalling to convey Layer 2 site reachability. This BGP signaling automates the mapping of remote Layer 2 VPN sites to LSP forwarding next hops. This means that with a Layer 2 VPN explicit mapping of an LSP to a PE device's edge-facing interface is not required.

For details on CCC, refer to Configuring an MPLS-Based VLAN CCC Using a Layer 2 Circuit.

Overview and Topology

A Layer 2 VPN provides complete separation between the provider and customer networks. The benefits of a Layer 2 VPN include support for nonstandard transport protocols and the isolation of link addressing and routing protocol operation between the customer and provider networks.

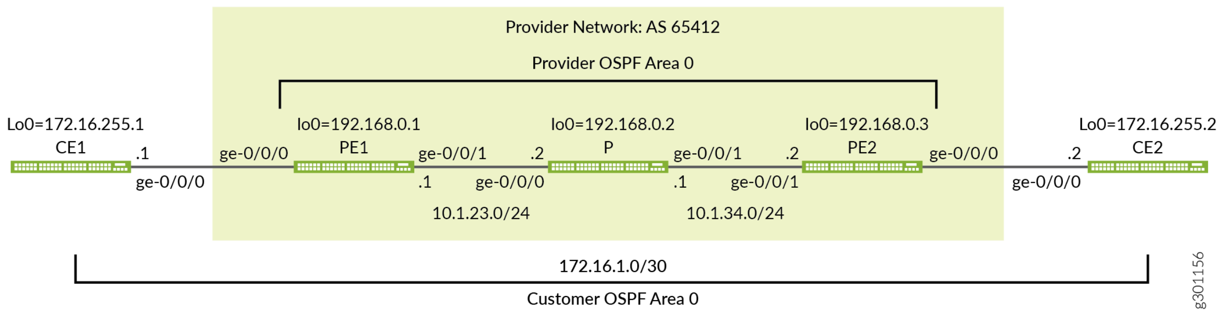
Definition of a VPN involves changes to the local and remote PE devices only. No additional configuration is needed on the provider devices (aside from baseline MPLS support), because these devices only provide basic MPLS switching functions. The CE devices do not use MPLS. They require only a basic interface, and if desired, protocol configuration, to operate over the Layer 2 VPN. For a Layer 2 VPN you configure the CE devices as if they were attached to a shared link.

Once an MPLS baseline is in place, you must configure the following functionality on the PE devices to establish an MPLS-based Layer 2 VPN:

- A BGP group with `family l2vpn signaling`
- A routing instance with instance type `l2vpn`
- The customer-facing interfaces on the PE devices must be configured as follows:
  - Specify `ethernet-ccc` or `vlan-ccc` physical layer encapsulation depending on whether VLAN tagging is in use.
  - Configure a matching encapsulation type in the routing instance configuration.
  - Configure the logical interface (unit) used for the Layer 2 VPN with `family ccc`.

Figure 13 on page 183 provides the topology for this MPLS-based Layer 2 VPN example. The figure details the interface names, IP addressing, and protocols used in the provider network. It also highlights the end-to-end nature of the CE device addressing and protocol stack operation. Unlike a Layer 3 VPN, CE device operation is opaque to the provider network in a Layer 2 VPN. There is no peering relationship between the CE devices and the provider network. As a result you expect the CE devices to form an OSPF adjacency *across*, *not to*, the provider network.

Figure 13: An MPLS-Based Layer 2 VPN



Quick Configurations

IN THIS SECTION

- CLI Quick Configuration | 183

Use the configurations in this section to quickly get your MPLS-based Layer 2 VPN up and running. The configurations include a functional MPLS baseline to support your Layer 2 VPN. This example focuses on the VPN aspects of the configuration. Refer to the following links for additional information on the baseline MPLS functionality used in this example:

- Configuring MPLS on Provider Edge EX8200 and EX4500 Switches Using Circuit Cross-Connect
- Configuring MPLS on EX8200 and EX4500 Provider Switches

CLI Quick Configuration

NOTE: The device configurations omit the management interface, static routes, system logging, system services, and user login information. These parts of the configuration vary by location and are not directly related to MPLS or VPN functionality.

Edit the following commands as needed for the specifics of your environment and paste them into the local CE (CE1) device terminal window:

The complete configuration for the CE1 device.

```
set system host-name ce1
set interfaces ge-0/0/0 description "Link from CE1 to PE1"
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.1/30
set interfaces lo0 unit 0 family inet address 172.16.255.1/32
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
```

Edit the following commands as needed for the specifics of your environment and paste them into the local PE (PE1) device terminal window:

The complete configuration for PE1 device.

```
set system host-name pe1
set interfaces ge-0/0/0 description "Link from PE1 to CE1"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc
set interfaces ge-0/0/1 description "Link from PE1 to P-router"
set interfaces ge-0/0/1 mtu 4000
set interfaces ge-0/0/1 unit 0 family inet address 10.1.23.1/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.0.1/32
set routing-instances l2vpn1 protocols l2vpn interface ge-0/0/0.0 description "EDGE LINK BETWEEN
PE1 AND CE1"
set routing-instances l2vpn1 protocols l2vpn site CE-1 interface ge-0/0/0.0 remote-site-id 2
set routing-instances l2vpn1 protocols l2vpn site CE-1 site-identifier 1
set routing-instances l2vpn1 protocols l2vpn encapsulation-type ethernet
set routing-instances l2vpn1 instance-type l2vpn
set routing-instances l2vpn1 interface ge-0/0/0.0
set routing-instances l2vpn1 route-distinguisher 192.168.0.1:12
set routing-instances l2vpn1 vrf-target target:65412:12
set routing-options autonomous-system 65412
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.168.0.1
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 192.168.0.3
set protocols mpls label-switched-path lsp_to_pe2 to 192.168.0.3
set protocols mpls interface ge-0/0/1.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols rsvp interface lo0.0
set protocols rsvp interface ge-0/0/1.0
```

The complete configuration for the P device.

```
set system host-name p
set interfaces ge-0/0/0 description "Link from P-router to PE1"
set interfaces ge-0/0/0 mtu 4000
set interfaces ge-0/0/0 unit 0 family inet address 10.1.23.2/24
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 description "Link from P-router to PE2"
set interfaces ge-0/0/1 mtu 4000
set interfaces ge-0/0/1 unit 0 family inet address 10.1.34.1/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface ge-0/0/1.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols rsvp interface lo0.0
set protocols rsvp interface ge-0/0/0.0
set protocols rsvp interface ge-0/0/1.0
```

The complete configuration for the PE2 device.

```
set system host-name pe2
set interfaces ge-0/0/0 description "Link from PE2 to CE2"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc
set interfaces ge-0/0/1 description "Link from PE2 to P-router"
set interfaces ge-0/0/1 mtu 4000
set interfaces ge-0/0/1 unit 0 family inet address 10.1.34.2/24
set interfaces ge-0/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set routing-instances l2vpn1 protocols l2vpn interface ge-0/0/0.0 description "EDGE LINK BETWEEN
PE2 AND CE2"
set routing-instances l2vpn1 protocols l2vpn site CE-2 interface ge-0/0/0.0 remote-site-id 1
set routing-instances l2vpn1 protocols l2vpn site CE-2 site-identifier 2
set routing-instances l2vpn1 protocols l2vpn encapsulation-type ethernet
set routing-instances l2vpn1 instance-type l2vpn
set routing-instances l2vpn1 interface ge-0/0/0.0
set routing-instances l2vpn1 route-distinguisher 192.168.0.3:12
set routing-instances l2vpn1 vrf-target target:65412:12
set routing-options autonomous-system 65412
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.168.0.3
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 192.168.0.1
set protocols mpls label-switched-path lsp_to_pe1 to 192.168.0.1
set protocols mpls interface ge-0/0/1.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols rsvp interface lo0.0
set protocols rsvp interface ge-0/0/1.0
```

The complete configuration for the CE2 device.

```
set system host-name ce2
set interfaces ge-0/0/0 description "Link from CE2 to PE2"
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/30
set interfaces lo0 unit 0 family inet address 172.16.255.2/32
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
```

Be sure to commit the configuration changes on all devices when satisfied with your work. Congratulations on your new MPLS-based Layer 2 VPN! Refer to the "Verification" on page 200 section for the steps needed to confirm your VPN is working as expected.

Configure the Local PE (PE1) Device for a MPLS-Based Layer 2 VPN

IN THIS SECTION

- Procedure | 188
- Results | 191

This section covers the steps needed to configure the PE1 device for this example. Refer to the "Example: Configure MPLS-Based Layer 2 VPNs" on page 180 section for the CE device and P device configurations used in this example.

Configure the MPLS Baseline (if Needed)

Before you configure the Layer 2 VPN make sure the PE device has a working MPLS baseline. If you already having a an MPLS baseline you can skip to the step-by-step procedure to add the Layer 2 VPN to the local PE device.

- Configure the hostname.

```
[edit]
user@pe1# set system host-name pe1
```

- Configure the interfaces.

```
[edit]
user@pe1# set interfaces ge-0/0/1 description "Link from PE1 to P-router"
[edit]
user@pe1# set interfaces ge-0/0/1 mtu 4000
[edit]
user@pe1# set interfaces ge-0/0/1 unit 0 family inet address 10.1.23.1/24
[edit]
user@pe1# set interfaces ge-0/0/1 unit 0 family mpls
[edit]
user@pe1# set interfaces lo0 unit 0 family inet address 192.168.0.1/32
```

CAUTION: Layer 2 VPNs don’t support fragmentation in the provider network. It is critical that the provider network supports the largest frame that the CE devices can generate *after* the MPLS and virtual routing and forwarding (VRF) labels are added by the PE devices. This example leaves the CE devices at the default 1500-byte maximum transmission unit (MTU) while configuring the provider core to support a 4000 byte MTU. This configuration avoids discards by ensuring the CE devices cannot exceed the MTU in the provider’s network.

- Configure the protocols.

NOTE: Traffic engineering is supported for RSVP-signaled LSPs but is not required for basic MPLS switching or VPN deployment. The provided MPLS baseline uses RSVP to signal LSPs, and enables traffic engineering for OSPF. However, no path constraints are configured so you expect the LSPs to be routed over the interior gateway protocol's shortest path.

```
[edit
]user@pe1# set protocols ospf area 0.0.0.0 interface lo0.0
[edit]
user@pe1# set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
[edit]
user@pe1# set protocols ospf traffic-engineering
[edit]
user@pe1# set protocols mpls interface ge-0/0/1.0
[edit]
user@pe1# set protocols rsvp interface lo0.0
[edit]
user@pe1# set protocols rsvp interface ge-0/0/1.0
```

- Define the LSP to the remote PE device's loopback address.

```
[edit]
user@pe1#  set protocols mpls label-switched-path lsp_to_pe2 to 192.168.0.3
```

Procedure

Step-by-Step Procedure

Follow these steps to configure the PE1 device for a Layer 2 VPN.

1. Configure the edge-facing interface. Specify a physical encapsulation type of `ethernet-ccc` with `family ccc` on unit 0. This is the only valid unit number for an untagged Ethernet interface. If you are using VLAN tagging specify `vlan-ccc` encapsulation and add the CCC family to the desired unit(s).

TIP: You can configure both an MPLS-based Layer 2 VPN and an MPLS-based Layer 3 VPN on the same PE device. However, you cannot configure the same customer edge-facing interface to support both a Layer 2 VPN and a Layer 3 VPN.

```
[edit]user@pe1# set interfaces ge-0/0/0 encapsulation ethernet-ccc
[edit]
user@pe1# set interfaces ge-0/0/0 unit 0 family ccc
[edit]
user@pe1# set interfaces ge-0/0/0 description "Link from PE1 to CE1"
```

NOTE: A Layer 2 VPN requires that the PE device's edge-facing interfaces be configured with CCC encapsulation at the physical device level with the CCC family configured at the unit level. The provider devices are configured in the same way whether you are deploying CCC, an MPLS-based Layer 2 VPN, or an MPLS-based Layer 3 VPN. This is because they have no edge-facing interfaces or VPN awareness.

2. Configure a BGP group for the peering between the local and remote PE devices. Use the PE device's loopback address as the local address and enable `family l2vpn signaling`.

```
[edit protocols bgp]
user@pe1# set group ibgp local-address 192.168.0.1 family l2vpn signaling
```

3. Configure the BGP group type as internal.

```
[edit protocols bgp]
user@pe1# set group ibgp type internal
```

4. Configure the remote PE device's loopback address as a BGP neighbor.

```
[edit protocols bgp]
user@pe1# set group ibgp neighbor 192.168.0.3
```

5. Configure the BGP autonomous system number.

```
[edit routing-options]
user@pe1# set autonomous-system 65412
```

6. Configure the routing instance. Start by specifying the instance name *l2vpn1*, with an `instance-type` of `l2vpn`.

```
[edit routing-instances]
user@pe1# set l2vpn1 instance-type l2vpn
```

7. Configure the PE device's customer-facing interface to belong to the routing instance.

```
[edit routing-instances]
user@pe1# set l2vpn1 interface ge-0/0/0
```

8. Configure the routing instance's route distinguisher. This setting is used to distinguish the routes sent from a particular VRF on a particular PE device. It should be unique for each routing instance on each PE device.

```
[edit routing-instances]
user@pe1# set l2vpn1 route-distinguisher 192.168.0.1:12
```

9. Configure the instance's virtual routing and forwarding (VRF) table route target. The `vrf-target` statement adds the specified community tag to all advertised routes while automatically matching the same value for route import. Configuring matching route targets on the PE devices that share a given VPN is required for proper route exchange.

```
[edit routing-instances]
user@pe1# set l2vpn1 vrf-target target:65412:12
```

NOTE: You can create more complex policies by explicitly configuring VRF import and export policies using the import and export options. See *vrf-import* and *vrf-export* for details.

10. Configure the l2vpn protocol in the instance and specify the encapsulation that is used on the edge-facing link. If the edge interface is VLAN tagged, be sure to specify ethernet-vlan.

    ```
    [edit routing-instances]
    user@pe1# set l2vpn1 protocols l2vpn encapsulation-type ethernet
    ```

11. Add the edge-facing interface under the instance's l2vpn stanza along with a description.

    ```
    [edit routing-instances]
    user@pe1# set l2vpn1 protocols l2vpn interface ge-0/0/0.0 description "L2vpn Link Between
    PE1 and CE1"
    ```

12. Configure the Layer 2 VPN site information and associate the edge-facing interface with the local customer site.

    ```
    [edit routing-instances]
    user@pe1# set l2vpn1 protocols l2vpn site CE-1 site-identifier 1 interface ge-0/0/0.0
    remote-site-id 2
    ```

    NOTE: In this example, the site ID for the PE1 device is *1* and the site ID for the PE2 device is *2*. For the local PE device (PE1), the remote site is correctly configured with a remote-site-id value of 2.

13. Commit your changes at the PE1 device and return to CLI operational mode.

    ```
    [edit]
    user@pe1# commit and-quit
    ```

Results

Display the results of the configuration on the PE1 device. The output reflects only the functional configuration added in this example.

```
user@pe1> show configuration
interfaces {
    ge-0/0/0 {
```

```
        description "Link from PE1 to CE1";
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/1 {
        description "Link from PE1 to P-router";
        mtu 4000;
        unit 0 {
            family inet {
                address 10.1.23.1/24;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.0.1/32;
            }
        }
    }
}
routing-instances {
    l2vpn1 {
        protocols {
            l2vpn {
                interface ge-0/0/0.0 {
                    description "L2vpn Link Between PE1 and CE1" ;
                }
                site CE-1 {
                    interface ge-0/0/0.0 {
                        remote-site-id 2;
                    }
                    site-identifier 1;
                }
                encapsulation-type ethernet;
            }
        }
        instance-type l2vpn;
        interface ge-0/0/0.0;
        route-distinguisher 192.168.0.1:12;
        vrf-target target:65412:12;
    }
}
routing-options {
    autonomous-system 65412;
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 192.168.0.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.168.0.3;
        }
    }
    mpls {
        label-switched-path lsp_to_pe2 {
            to 192.168.0.3;
        }
        interface ge-0/0/1.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0;
            interface ge-0/0/1.0;
        }
    }
    rsvp {
        interface lo0.0;
        interface ge-0/0/1.0;
    }
}
```

Configure the Remote PE (PE2) Device for a MPLS-Based Layer 2 VPN

IN THIS SECTION

- Procedure | 195

This section covers the steps needed to configure the PE2 device for this example. Refer to the "Example: Configure MPLS-Based Layer 2 VPNs" on page 180 section for the CE device and P device configurations used in this example.

Configure the MPLS Baseline (if Needed)

Before you configure the Layer 2 VPN make sure the PE device has a working MPLS baseline. If you already having an MPLS baseline you can skip to the step-by-step procedure to add the Layer 2 VPN to the local PE device.

- Configure the hostname.

```
[edit]
user@pe2# set system host-name pe2
```

- Configure the interfaces.

```
[edit]
user@pe2# set interfaces ge-0/0/1 description "Link from PE2 to P-router"
[edit]
user@pe2# set interfaces ge-0/0/1 mtu 4000
[edit]
user@pe2# set interfaces ge-0/0/1 unit 0 family inet address 10.1.34.2/24
[edit]
user@pe2# set interfaces ge-0/0/1 unit 0 family mpls
[edit]
user@pe2# set interfaces lo0 unit 0 family inet address 192.168.0.3/32
```

CAUTION: Layer 2 VPNs don’t support fragmentation in the provider network. It is critical that the provider network supports the largest frame that the CE devices can generate *after* the MPLS and virtual routing and forwarding (VRF) labels are added by the PE devices. This example leaves the CE devices at the default 1500-byte maximum transmission unit (MTU) while configuring the provider core to support a 4000 byte MTU. This configuration avoids discards by ensuring the CE devices cannot exceed the MTU in the provider's network.

- Configure the protocols.

NOTE: Traffic engineering is supported for RSVP-signaled LSPs but is not required for basic MPLS switching or VPN deployment. The provided MPLS baseline uses RSVP to signal LSPs, and enables traffic engineering for OSPF. However, no path constraints are configured so you expect the LSPs to be routed over the interior gateway protocol's shortest path.

```
[edit]
user@pe2# set protocols ospf area 0.0.0.0 interface lo0.0
[edit]
user@pe2# set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
[edit]
user@pe2# set protocols ospf traffic-engineering
[edit]
user@pe2# set protocols mpls interface ge-0/0/1.0
[edit]
user@pe2# set protocols rsvp interface lo0.0
[edit]
user@pe2# set protocols rsvp interface ge-0/0/1.0
```

- Define the LSP to the remote PE device's loopback address.

```
[edit]
user@pe2#  set protocols mpls label-switched-path lsp_to_pe1 to 192.168.0.1
```

Procedure

Step-by-Step Procedure

Follow these steps to configure the PE2 device for a Layer 2 VPN.

1. Configure the edge-facing interface encapsulation and family. Recall this is an untagged interface, therefore only unit 0 is valid for the `ccc` family.

```
[edit]user@pe2# set interfaces ge-0/0/0 encapsulation ethernet-ccc
[edit]
user@pe2# set interfaces ge-0/0/0 unit 0 family ccc
[edit]
user@pe1# set interfaces ge-0/0/0 description "Link from PE2 to CE2"
```

2. Configure a BGP group. Specify the PE device's loopback address as the local address and enable `family l2vpn signaling`.

```
[edit protocols bgp]
user@pe2# set group ibgp local-address 192.168.0.3 family l2vpn signaling
```

3. Configure the BGP group type as internal.

```
[edit protocols bgp]
user@pe2# set group ibgp type internal
```

4. Configure the PE1 device as a BGP neighbor. Be sure to specify PE1's loopback address as the BGP neighbor.

```
[edit protocols bgp]
user@pe2# set group ibgp neighbor 192.168.0.1
```

5. Configure the BGP autonomous system number.

```
[edit routing-options]
user@pe2# set autonomous-system 65412
```

6. Configure the routing instance. Start by specifying the instance name *l2vpn1* with an `instance-type` of `l2vpn`.

```
[edit routing-instances]
user@pe2# set l2vpn1 instance-type l2vpn
```

7. Configure the PE device’s customer edge-facing interface to belong to the routing instance.

```
[edit routing-instances]
user@pe2# set l2vpn1 interface ge-0/0/0
```

8. Configure the instance’s route distinguisher.

```
[edit routing-instances]
user@pe2# set l2vpn1 route-distinguisher 192.168.0.3:12
```

9. Configure the instance’s VPN virtual routing and forwarding (VRF) table route target. The assigned target must match the one configured at the PE1 device.

```
[edit routing-instances]
user@pe2# set l2vpn1 vrf-target target:65412:12
```

10. Configure the instance for the `l2vpn` protocol and specify the encapsulation used on the edge-facing link.

```
[edit routing-instances]
user@pe2# set l2vpn1 protocols l2vpn encapsulation-type ethernet
```

11. Add the PE device’s edge-facing interface under the instance’s `l2vpn` hierarchy along with a description .

```
[edit routing-instances]
user@pe2# set l2vpn1 protocols l2vpn interface ge-0/0/0.0 description "L2vpn Link Between
PE2 and CE2"
```

12. Configure the instance’s Layer 2 VPN site information and list the PE device's edge-facing interface under the local site. The local site ID configured on the PE2 device must match the remote site ID you configured on the PE1 device, and vice versa.

```
[edit routing-instances]
user@pe1# set l2vpn1 protocols l2vpn site CE-2 site-identifier 2 interface ge-0/0/0.0
remote-site-id 1
```

NOTE: In this example, the site ID for the PE2 device is *2* and the site ID for the PE1 device is *1*. For the PE2 device the remote site is correctly configured with a `remote-site-id` value of 1.

13. Commit your changes at the PE2 device and return to CLI operational mode.

```
[edit]
user@pe1# commit and-quit
```

Results

Display the results of the configuration on the PE2 device.

```
user@pe2# show
```

```
interfaces {
    ge-0/0/0 {
        description "Link from PE2 to CE2";
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/1 {
        description "Link from PE2 to P-router";
        mtu 4000;
        unit 0 {
            family inet {
                address 10.1.34.2/24;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.0.3/32;
            }
        }
    }
}
routing-instances {
    l2vpn1 {
        protocols {
            l2vpn {
                interface ge-0/0/0.0 {
                    description "L2vpn Link Between PE2 and CE2" ;
                }
                site CE-2 {
                    interface ge-0/0/0.0 {
                        remote-site-id 1;
                    }
                    site-identifier 2;
                }
                encapsulation-type ethernet;
            }
        }
        instance-type l2vpn;
        interface ge-0/0/0.0;
        route-distinguisher 192.168.0.3:12;
        vrf-target target:65412:12;
    }
}
routing-options {
    autonomous-system 65412;
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 192.168.0.3;
            family l2vpn {
                signaling;
            }
            neighbor 192.168.0.1;
        }
    }
    mpls {
        label-switched-path lsp_to_pe1 {
            to 192.168.0.1;
        }
        interface ge-0/0/1.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0;
            interface ge-0/0/1.0;
        }
    }
    rsvp {
        interface lo0.0;
        interface ge-0/0/1.0;
    }
}
```

Verification

IN THIS SECTION

- Verify Provider OSPF Adjacencies and Route Exchange | **200**
- Verify MPLS and RSVP Interface Settings | **201**
- Verify RSVP Signaled LSPs | **202**
- Verify BGP Session Status | **203**
- Verify Layer 2 VPN Routes in the Routing Table | **203**
- Verify Layer 2 VPN Connection Status | **205**
- Ping the Remote PE Device Using the Layer 2 VPN Connection | **206**
- Verify End-to-End Operation of the CE Devices Over the Layer 2 VPN | **208**

Perform these tasks to verify that the MPLS-based Layer 2 VPN works properly:

Verify Provider OSPF Adjacencies and Route Exchange

Purpose

Confirm the OSPF protocol is working properly in the provider network by verifying adjacency status and OSPF learned routes to the loopback addresses of the remote provider devices. Proper IGP operation is critical for the successful establishment of MPLS LSPs.

Action

```
user@pe1> show ospf neighbor

Address          Interface              State           ID               Pri  Dead
10.1.23.2        ge-0/0/1.0             Full            192.168.0.2      128    38
```

```
user@pe1> show route protocol ospf | match 192.168

192.168.0.2/32     *[OSPF/10] 1w5d 20:48:59, metric 1
192.168.0.3/32     *[OSPF/10] 2w0d 00:08:30, metric 2
```

Meaning

The output shows that the PE1 device has established an OSPF adjacency to the P device (`192.168.0.2`). It also shows that the P and remote PE device loopback addresses (`192.168.0.2`) and (`192.168.0.3`) are learned via OSPF at the local PE device.

Verify MPLS and RSVP Interface Settings

Purpose

Verify that the RSVP and MPLS protocols are configured to operate on the PE device's core-facing interfaces. This step also verifies that `family mpls` is correctly configured at the unit level of the core-facing interfaces.

Action

```
user@pe1> show mpls interface

Interface        State       Administrative groups (x: extended)
ge-0/0/1.0       Up         <none>
```

```
user@pe1> show rsvp interface

RSVP interface: 2 active
                          Active  Subscr- Static      Available   Reserved    Highwater
```

```
Interface          State  resv    iption  BW          BW          BW          mark
ge-0/0/1.0             Up       1   100%  1000Mbps    1000Mbps    0bps        0bps
lo0.0                  Up       0   100%  0bps        0bps        0bps        0bps
```

Meaning

The output shows that MPLS and RSVP are correctly configured on the local PE device's core-facing and loopback interfaces.

Verify RSVP Signaled LSPs

Purpose

Verify that the RSVP sessions (ingress and egress) are properly established between the PE devices.

Action

```
user@pe1> show rsvp session

To              From            State   Rt Style Labelin Labelout LSPname
192.168.0.3     192.168.0.1     Up       0  1 FF       -   299888 lsp_to_pe2
Total 1 displayed, Up 1, Down 0

Egress RSVP: 1 sessions
To              From            State   Rt Style Labelin Labelout LSPname
192.168.0.1     192.168.0.3     Up       0  1 FF       3        - lsp_to_pe1
Total 1 displayed, Up 1, Down 0

Transit RSVP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

Meaning

The output shows that both the ingress and egress RSVP sessions are correctly established between the PE devices. Successful LSP establishment indicates the MPLS baseline is operational.

Verify BGP Session Status

Purpose

Verify that the BGP session between the PE devices is correctly established with support for Layer 2 VPN network layer reachability information (NLRI).

Action

```
user@pe1> show bgp summary
Threading mode: BGP I/O
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0
                       1          1          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.0.3           65412          6          5       0       0        1:34 Establ
  bgp.l2vpn.0: 1/1/1/0
  l2vpn1.l2vpn.0: 1/1/1/0
```

Meaning

The output shows the BGP session to the remote PE device (`192.168.0.3`) has been correctly established (`Establ`), and through the `Up/Dwn` field, how long the session has been in the current state (`1:34`). It also shows the number of BGP packets sent to (`5`) and received from (`6`) the remote PE device. The `flaps` field confirms that no state transitions have occurred (`0`), indicating the session is stable. Also note that Layer 2 VPN NLRI is correctly exchanged between the PE devices. This output confirms the BGP peering between the PE devices is ready to support a Layer 2 VPN.

Verify Layer 2 VPN Routes in the Routing Table

Purpose

Verify that the routing table on the PE1 device is populated with the Layer 2 VPN routes used to forward traffic between the CE devices.

Action

```
user@pe1> show route table bgp.l2vpn.0

bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.3:12:2:1/96
                   *[BGP/170] 00:51:36, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    >  to 10.1.23.2 via ge-0/0/1.0, label-switched-path lsp_to_pe2
```

```
user@pe1> show route table l2vpn1.l2vpn.0

l2vpn1.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.0.1:12:1:1/96
                   *[L2VPN/170/-101] 01:48:30, metric2 1
                       Indirect
192.168.0.3:12:2:1/96
                   *[BGP/170] 00:51:57, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    >  to 10.1.23.2 via ge-0/0/1.0, label-switched-path lsp_to_pe2
```

Meaning

The command `show route table bgp.l2vpn.0` displays all Layer 2 VPN routes that have been received on the PE device. The command `show route table l2vpn1.l2vpn.0` shows the Layer 2 VPN routes that have been imported into the `l2vpn1` routing instance as a result of a matching route target. The `l2vpn1.l2vpn.0` table contains both the local PE device's Layer 2 VPN route as well as a remote route learned via the BGP peering to the remote PE device. Both tables show the remote Layer 2 VPN route is correctly associated with the `lsp_to_pe2` LSP as a forwarding next hop. The outputs confirm the local PE device has learned about the remote customer site from the PE2 device. It also shows that it can forward Layer 2 VPN traffic to the PE2 device using MPLS transport over the provider network.

Verify Layer 2 VPN Connection Status

Purpose

Verify the status of the Layer 2 VPN connection.

Action

```
user@pe1> show l2vpn connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch           HS -- Hot-standby Connection

Legend for interface status
Up -- operational
Dn -- down

Instance: l2vpn1
Edge protection: Not-Primary
  Local site: CE-1 (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Jul 28 10:47:18 2020           1
      Remote PE: 192.168.0.3, Negotiated control-word: Yes (Null)
```

```
        Incoming label: 800009, Outgoing label: 800006
        Local interface: ge-0/0/0.0, Status: Up, Encapsulation: ETHERNET
        Flow Label Transmit: No, Flow Label Receive: No
```

Meaning

The `St` field in the output shows that the Layer 2 VPN connection to `Remote PE 192.168.0.3` at `connection-site 2` is `Up`. The output also confirms the PE device's edge-facing interface name `ge-0/0/0.0` and operational status as `up`. You also verify that Ethernet encapsulation is configured on the PE device's customer-facing interface. This is the correct encapsulation for the untagged Ethernet interfaces used in this example. The verification steps performed thus far indicate that the Layer 2 VPN's control plane is operational. You verify the data plane of the Layer 2 VPN in the following steps.

Ping the Remote PE Device Using the Layer 2 VPN Connection

Purpose

Verify Layer 2 VPN connectivity between the local and remote PE devices. Two forms of the `ping mpls l2vpn` command are shown. Both test Layer 2 VPN routing and MPLS forwarding between the PE devices. The first command assumes a single remote site while the second specifies the local and remote site identifiers, which is useful when testing a multi-site Layer 2 VPN. This is because the remote site ID can be used to target the desired remote PE device.

NOTE: The `ping mpls l2vpn` command validates Layer 2 VPN route exchange and MPLS forwarding between the PE devices. This is done by generating traffic from the local PE's Layer 2 VPN routing instance to the remote PE device's 127.0.0.1 loopback address. This command does not validate the operation of the CE device interfaces or their configuration. This is because CE device operation is opaque to the provider network in a Layer 2 VPN.

Action

```
user@pe1> ping mpls l2vpn interface ge-0/0/0.0 reply-mode ip-udp

!!!!!
```

```
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

```
user@pe1> ping mpls l2vpn instance l2vpn1 remote-site-id 2 local-site-id 1 detail

Request for seq 1, to interface 334, labels <800002, 299840>, packet size 88
Reply for seq 1, return code: Egress-ok, time: 593.784 ms
        Local transmit time: 2020-07-13 16:15:55 UTC 241.357 ms
        Remote receive time: 2020-07-13 16:15:55 UTC 835.141 ms
Request for seq 2, to interface 334, labels <800002, 299840>, packet size 88
Reply for seq 2, return code: Egress-ok, time: 591.700 ms
        Local transmit time: 2020-07-13 16:15:56 UTC 241.405 ms
        Remote receive time: 2020-07-13 16:15:56 UTC 833.105 ms
Request for seq 3, to interface 334, labels <800002, 299840>, packet size 88
Reply for seq 3, return code: Egress-ok, time: 626.084 ms
        Local transmit time: 2020-07-13 16:15:57 UTC 241.407 ms
        Remote receive time: 2020-07-13 16:15:57 UTC 867.491 ms
Request for seq 4, to interface 334, labels <800002, 299840>, packet size 88
Reply for seq 4, return code: Egress-ok, time: 593.061 ms
        Local transmit time: 2020-07-13 16:15:58 UTC 241.613 ms
        Remote receive time: 2020-07-13 16:15:58 UTC 834.674 ms
Request for seq 5, to interface 334, labels <800002, 299840>, packet size 88
Reply for seq 5, return code: Egress-ok, time: 594.192 ms
        Local transmit time: 2020-07-13 16:15:59 UTC 241.357 ms
        Remote receive time: 2020-07-13 16:15:59 UTC 835.549 ms

--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Meaning

The output confirms that the Layer 2 VPN forwarding plane is operating correctly between the PE devices.

Verify End-to-End Operation of the CE Devices Over the Layer 2 VPN

Purpose

Verify Layer 2 VPN connectivity between the CE devices. This step confirms the CE devices have operational interfaces and are properly configured for Layer 2 connectivity. This is done by verifying the CE devices have established an OSPF adjacency and are able to pass traffic end-to-end between their loopback addresses.

Action

```
user@ce1> show ospf neighbor

Address          Interface              State           ID               Pri  Dead
172.16.1.2       ge-0/0/0.0             Full            172.16.255.2     128    32
```

```
user@ce1> show ospf route | match 172

172.16.255.2/32    *[OSPF/10] 01:34:50, metric 1
                    >  to 172.16.1.2 via ge-0/0/0.0
```

```
user@ce1> ping 172.16.255.2 size 1472 do-not-fragment count 2

PING 172.16.255.2 (172.16.255.2): 1472 data bytes
1480 bytes from 172.16.255.2: icmp_seq=0 ttl=64 time=4.404 ms
1480 bytes from 172.16.255.2: icmp_seq=1 ttl=64 time=5.807 ms

--- 172.16.255.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 4.404/5.106/5.807/0.702 ms
```

Meaning

The output shows that Layer 2 VPN connectivity is working correctly between the CE devices. It confirms that the local CE device has established an OSPF adjacency over the provider core to the remote CE device `172.16.1.2`, and that the local CE device has learned a route to the remote CE device's loopback address `172.16.255.2` via OSPF. The output also shows that the CE devices are able to pass

1500-byte IP packets without evoking local fragmentation. The successful pings also verify the frames did not exceed the MTU supported by the provider’s network.

NOTE: The `size` argument added to the `ping` command generates 1472 bytes of echo data. An additional 8 bytes of Internet Control Message Protocol (ICMP) and 20 bytes of IP header are added to bring the total packet size to 1500-bytes. Adding the `do-not-fragment` switch ensures the CE device cannot perform fragmentation based on its local MTU. This method confirms that no fragmentation is possible, or needed, when sending standard length Ethernet frames between the CE devices.

RELATED DOCUMENTATION

Example: Configuring MPLS on EX8200 and EX4500 Switches

Example: Configure a Basic MPLS-Based Layer 3 VPN

Transmitting Nonstandard BPDUs in Layer 2 VPNs and VPLS

Circuit cross-connect (CCC) protocol, Layer 2 circuit, and Layer 2 VPN configurations can transmit nonstandard bridge protocol data units (BPDUs) generated by other vendors’ equipment. This is the default behavior on all supported PICs and requires no additional configuration.

CHAPTER 11

Configuring Layer 2 Interfaces

IN THIS CHAPTER

- Configuring CCC Encapsulation for Layer 2 VPNs | **210**
- Configuring TCC Encapsulation for Layer 2 VPNs and Layer 2 Circuits | **211**
- Configuring the MTU for Layer 2 Interfaces | **213**
- Disabling the Control Word for Layer 2 VPNs | **214**

Configuring CCC Encapsulation for Layer 2 VPNs

You need to specify a circuit cross-connect (CCC) encapsulation type for each PE-router-to-CE-router interface running a Layer 2 VPN. This encapsulation type should match the encapsulation type configured under the routing instance. For information about how to configure the encapsulation type under the routing instance, see "Configuring the Encapsulation Type" on page 154 .

NOTE: A Layer 2 VPN or Layer 2 circuit is not supported if the PE-router-to-P-router interface has VLAN-tagging enabled and uses a nonenhanced Flexible PIC Concentrator (FPC).

For Layer 2 VPNs, you need to configure the CCC encapsulation on the logical interface. You also need to configure an encapsulation on the physical interface. The physical interface encapsulation does not have to be a CCC encapsulation. However, it should match the logical interface encapsulation. For example, if you configure an Ethernet CCC encapsulation type on the logical interface, you should configure a compatible Ethernet encapsulation on the physical interface.

To configure the CCC encapsulation type, include the `encapsulation-type` statement:

```
encapsulation-type ccc-encapsulation-type;
```

To configure the CCC encapsulation type on the physical interface, include this statement at the following hierarchy levels:

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

To configure the CCC encapsulation type on the logical interface, include this statement at the following hierarchy levels:

- `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`*`]`
- `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logical-unit-number`*`]`

You configure the encapsulation type at the `[edit interfaces]` hierarchy level differently from the `[edit routing-instances]` hierarchy level. For example, you specify the encapsulation as `frame-relay` at the `[edit routing-instances]` hierarchy level and as `frame-relay-ccc` at the `[edit interfaces]` hierarchy level.

You can run both standard Frame Relay and CCC Frame Relay on the same device. If you specify Frame Relay encapsulation (`frame-relay-ccc`) for the interface, you should also configure the encapsulation at the `[edit interfaces` *`interface name`* `unit` *`unit-number`*`]` hierarchy level as `frame-relay-ccc`. Otherwise, the logical interface unit defaults to standard Frame Relay.

Configuring TCC Encapsulation for Layer 2 VPNs and Layer 2 Circuits

Also known as Layer 2.5 VPNs, the translation cross-connect (TCC) encapsulation types allow you to configure different encapsulation types at the ingress and egress of a Layer 2 VPN or the ingress and egress of a Layer 2 circuit. For example, a CE router at the ingress of a Layer 2 VPN path can send traffic in a Frame Relay encapsulation. A CE router at the egress of that path can receive the traffic in an ATM encapsulation.

NOTE: The EX9200 switches only use `ethernet` and `ethernet-vlan` encapsulation types.

For information about how to configure encapsulations for Layer 2 circuits, see "Configuring the Interface Encapsulation Type for Layer 2 Circuits" on page 366

The configuration for TCC encapsulation types is similar to the configuration for CCC encapsulation types. For Layer 2 VPNs, you specify a TCC encapsulation type for each PE-router-to-CE-router interface. The encapsulation type configured for the interface should match the encapsulation type configured under the routing instance. For information about how to configure the encapsulation type under the routing instance, see "Configuring the Encapsulation Type" on page 154.

NOTE: Some platform and FPC combinations can not pass TCC encapsulated ISO traffic. See Platforms/FPCs That Cannot Forward TCC Encapsulated ISO Traffic for details.

You need to configure the TCC encapsulation on both the physical and logical interfaces. To configure the TCC encapsulation type, include the `encapsulation-type` statement:

```
encapsulation-type tcc-encapsulation-type;
```

On the EX9200 switches, replace `encapsulation-type` with the `encapsulation` statement:

```
encapsulation tcc-encapsulation;
```

To configure the TCC encapsulation type on the physical interface, include this statement at the following hierarchy levels:

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

To configure the TCC encapsulation type on the logical interface, include this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number]`
- `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number]`

You configure the encapsulation type at the `[edit interfaces]` hierarchy level differently than at the `[edit routing-instances]` hierarchy level. For example, you specify the encapsulation as `frame-relay` at the `[edit routing-instances]` hierarchy level and as `frame-relay-tcc` at the `[edit interfaces]` hierarchy level.

For Layer 2.5 VPNs employing an Ethernet interface as the TCC router, you can configure an Ethernet TCC or an extended VLAN TCC.

To configure an Ethernet TCC or an extended VLAN TCC, include the `proxy` and `remote` statements:

```
proxy inet-address;
remote (inet-address | mac-address);
```

You can include these statements at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number family tcc]`

- `[edit logical-interfaces` *`logical-interface-name`* `interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family tcc]`

The `proxy inet-address` address statement defines the IP address for which the TCC router is acting as proxy.

The `remote (inet-address | mac-address)` statement defines the location of the remote router.

Ethernet TCC is supported on interfaces that carry IP version 4 (IPv4) traffic only. However, Ethernet TCC encapsulation is not supported on 8-port, 12-port, and 48-port Fast Ethernet PICs.

Configuring the MTU for Layer 2 Interfaces

By default, the MTU used to advertise a Layer 2 pseudowire is determined by taking the interface MTU for the associated physical interface and subtracting the encapsulation overhead for sending IP packets based on the encapsulation. However, encapsulations that support multiple logical interfaces (and multiple Layer 2 pseudowires) rely on the same interface MTU (since they are all associated with the same physical interface). This can prove to be a limitation for VLAN Layer 2 pseudowires using the same Ethernet interface.

This can also affect multivendor environments. For example, if you have three PE devices supplied by different vendors and one of the devices only supports an MTU of 1500, even if the other devices support larger MTUs you must configure the MTU as 1500 (the smallest MTU of the three PE devices).

You can explicitly configure which MTU is advertised for a Layer 2 pseudowire, even if the Layer 2 pseudowire is sharing a physical interface with other Layer pseudowires. When you explicitly configure an MTU for a Layer 2 pseudowire, be aware of the following:

- For BGP-based applications such as l2vpn, the advertised MTU will be zero unless an MTU value is explicitly set at the `[edit routing-instances` *`routing-instance-name`* `protocols (`*`l2vpn`* `) site` *`site-name`*`]` hierarchy level.

- An explicitly configured MTU is signaled to the remote PE device. The configured MTU is also compared to the MTU received from the remote PE device. If there is a conflict, the Layer 2 pseudowire is taken down.

- A configured MTU is used only in the control plane. It is not enforced in the data plane. You need to ensure that the CE device for a given Layer 2 pseudowire uses the correct MTU for data transmission.

The following procedure describes how to configure the MTU for the Layer 2 interface. This information applies to the following Layer 2 technologies:

- Layer 2 VPNs

- Layer 2 Circuits

1. To configure the MTU for a Layer 2 circuit, include the `mtu` statement:

   ```
   mtu mtu-number;
   ```

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

2. To allow a Layer 2 pseudowire to be established even though the MTU configured on the local PE router does not match the MTU configured on the remote PE router, include the `ignore-mtu-mismatch` statement:

   ```
   ignore-mtu-mismatch;
   ```

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

RELATED DOCUMENTATION

ignore-mtu-mismatch

mtu

Disabling the Control Word for Layer 2 VPNs

A 4-byte control word provides support for the emulated VC encapsulation for Layer 2 VPNs. This control word is added between the Layer 2 protocol data unit (PDU) being transported and the VC label that is used for demultiplexing. Various networking formats use the control word in a variety of ways.

On networks with equipment that does not support the control word, you can disable it by including the `no-control-word` statement:

```
no-control-word;
```

You can include this statement at 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]`

For more information about configuring the control word, see "Configuring the Control Word for Layer 2 Circuits" on page 357 and the *Layer 2 Circuits User Guide*.

RELATED DOCUMENTATION

Configuring Interfaces for Layer 2 Circuits | **357**

control-word

l2vpn

CHAPTER 12

Configuring Path Selection for Layer 2 VPNs and VPLS

IN THIS CHAPTER

- Understanding BGP Path Selection | **216**
- Enabling BGP Path Selection for Layer 2 VPNs and VPLS | **221**

Understanding BGP Path Selection

IN THIS SECTION

- Routing Table Path Selection | **218**
- BGP Table path selection | **220**
- Effects of Advertising Multiple Paths to a Destination | **220**

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

1. Verify that the next hop can be resolved.

2. Choose the path with the lowest preference value (routing protocol process preference).

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

3. Prefer the path with higher local preference.

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

4. If the accumulated interior gateway protocol (AIGP) attribute is enabled, add the IGP metric and prefer the path with the lower AIGP attribute.

5. Prefer the path with the shortest autonomous system (AS) path value (skipped if the `as-path-ignore` statement is configured).

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

6. Prefer the route with the lower origin code.

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

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

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

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

   - If nondeterministic routing table path selection behavior is configured (that is, the `path-selection cisco-nondeterministic` statement is included in the BGP configuration), prefer the path with the lowest MED metric.

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

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

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

8. Prefer strictly internal paths, which include IGP routes and locally generated routes (static, direct, local, and so forth).

9. Prefer strictly external BGP (EBGP) paths over external paths learned through internal BGP (IBGP) sessions.

10. Prefer the path whose next hop is resolved through the IGP route with the lowest metric. BGP routes that are resolved through IGP are preferred over unreachable or rejected routes.

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

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

11. If both paths are external, prefer the oldest path, in other words, the path that was learned first. This is done to minimize route-flapping. This rule is not used if any one of the following conditions is true:

    - **path-selection external-router-id** is configured.
    - Both peers have the same router ID.
    - Either peer is a confederation peer.
    - Neither path is the current active path.

12. Prefer the path from the peer with the lowest router ID. For any path with an originator ID attribute, substitute the originator ID for the router ID during router ID comparison.

13. Prefer the path with the shortest cluster list length. The length is 0 for no list.

14. Prefer the path from the peer with the lowest peer IP address.

15. Prefer a primary route over a secondary route. A primary route is one that belongs to the routing table. A secondary route is one that is added to the routing table through an export policy.

Routing Table Path Selection

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

NOTE: Starting with Junos OS Release 14.1R8, 14.2R7, 15.1R4, 15.1F6, and 16.1R1, the **as-path-ignore** option is supported for routing instances.

The routing process path selection takes place before BGP hands off the path to the routing table to makes its decision. To configure routing table path selection behavior, include the `path-selection` statement:

```
path-selection {
    (always-compare-med | cisco-non-deterministic | external-router-id);
    as-path-ignore;
    l2vpn-use-bgp-rules;
    med-plus-igp {
        igp-multiplier number;
        med-multiplier number;
    }
}
```

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

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

- Emulate the Cisco IOS default behavior (**cisco-non-deterministic**). This mode evaluates routes in the order that they are received and does not group them according to their neighboring AS. With `cisco-non-deterministic` mode, the active path is always first. All inactive, but eligible, paths follow the active path and are maintained in the order in which they were received, with the most recent path first. Ineligible paths remain at the end of the list.

  As an example, suppose you have three path advertisements for the 192.168.1.0 /24 route:

  - Path 1—learned through EBGP; AS Path of 65010; MED of 200
  - Path 2—learned through IBGP; AS Path of 65020; MED of 150; IGP cost of 5
  - Path 3—learned through IBGP; AS Path of 65010; MED of 100; IGP cost of 10

  These advertisements are received in quick succession, within a second, in the order listed. Path 3 is received most recently, so the routing device compares it against path 2, the next most recent advertisement. The cost to the IBGP peer is better for path 2, so the routing device eliminates path 3 from contention. When comparing paths 1 and 2, the routing device prefers path 1 because it is received from an EBGP peer. This allows the routing device to install path 1 as the active path for the route.

  NOTE: We do not recommend using this configuration option in your network. It is provided solely for interoperability to allow all routing devices in the network to make consistent route selections.

- Always comparing MEDs whether or not the peer ASs of the compared routes are the same (**always-compare-med**).
- Override the rule that If both paths are external, the currently active path is preferred (**external-router-id**). Continue with the next step (Step "15" on page 218) in the path-selection process.
- Adding the IGP cost to the next-hop destination to the MED value before comparing MED values for path selection (`med-plus-igp`).

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

BGP Table path selection

The following parameters are followed for BGP's path selection:

1. Prefer the highest local-preference value.
2. Prefer the shortest AS-path length.
3. Prefer the lowest origin value.
4. Prefer the lowest MED value.
5. Prefer routes learned from an EBGP peer over an IBGP peer.
6. Prefer best exit from AS.
7. For EBGP-received routes, prefer the current active route.
8. Prefer routes from the peer with the lowest Router ID.
9. Prefer paths with the shortest cluster length.
10. Prefer routes from the peer with the lowest peer IP address. Steps 2, 6 and 12 are the RPD criteria.

Effects of Advertising Multiple Paths to a Destination

BGP advertises only the active path, unless you configure BGP to advertise multiple paths to a destination.

Suppose a routing device has in its routing table four paths to a destination and is configured to advertise up to three paths (**add-path send path-count 3**). The three paths are chosen based on path selection criteria. That is, the three best paths are chosen in path-selection order. The best path is the active path. This path is removed from consideration and a new best path is chosen. This process is repeated until the specified number of paths is reached.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.1R8	Starting with Junos OS Release 14.1R8, 14.2R7, 15.1R4, 15.1F6, and 16.1R1, the **as-path-ignore** option is supported for routing instances.

RELATED DOCUMENTATION

Example: Ignoring the AS Path Attribute When Selecting the Best Path

Examples: Configuring BGP MED

Example: Advertising Multiple BGP Paths to a Destination

Enabling BGP Path Selection for Layer 2 VPNs and VPLS

Layer 2 VPNs and VPLS share the same path selection process for determining the optimal path to reach all of the destinations shared within a single routing instance. For Layer 2 VPN and VPLS topologies, the path selection process is straightforward if there is just a single path from each PE router to each CE device. However, the path selection process becomes more complex if the PE routers receive two or more valid paths to reach a specific CE device.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The following network scenarios provide examples of what might cause a PE router to receive more than one valid path to reach a specific CE device:

- Multihoming—One or more CE devices within a routing instance are multihomed to two or more PE routers. Each multihomed CE device has at least two valid paths.
- Route reflectors—There are multiple route reflectors deployed within the same network and they are supporting PE routers within the same routing instance. Due to time delays in large complex networks, the route reflectors can separately receive a different valid path to reach a CE device at different times. When they readvertise these valid paths, a PE router could receive two or more separate but apparently valid paths to the same CE device.

By default, Juniper Networks routers use just the designated forwarder path selection algorithm to select the best path to reach the Layer 2 VPN or VPLS destination (for more information, see "VPLS Path Selection Process for PE Routers" on page 753). However, you can also configure the routers in your network to use both the BGP path selection algorithm and the designated forwarder path selection algorithm as follows:

- On the Provider routers within the service providers network, the standard BGP path selection algorithm is used (for more information, see *Understanding BGP Path Selection*). Using the standard BGP path selection for Layer 2 VPN and VPLS routes allows a service provider to leverage the existing Layer 3 VPN network infrastructure to also support Layer 2 VPNs and VPLS. The BGP path selection algorithm also helps to ensure that the service provider's network behaves predictably with regard to Layer 2 VPN and VPLS path selection. This is particularly important in networks employing route reflectors and multihoming.

  When a Provider router receives multiple paths for the same destination prefix (for example, a multihomed CE device), one path is selected based on the BGP path selection algorithm and placed in the bgp.l2vpn.0 routing table and the appropriate *instance*.l2vpn.0 routing table.

- When a PE router receives all of the available paths to each CE device, it runs the designated forwarder path selection algorithm to select the preferred path to reach each CE device, independently of the results of the earlier BGP path selection algorithm run on the Provider router. The VPLS designated forwarder algorithm uses the D-bit, preference, and PE router identifier to determine which of the valid paths to each CE device to use. The PE router might select a path to reach a CE device which is different from the path selected by the BGP-based Provider routers. In this scenario, the following is the expected behavior for traffic sent to the multihomed CE device:

  - If the path selected by the remote PE router is available, traffic will traverse the network to the multihomed CE device using the remote PE router's preferred path (again, ignoring the path selected by the BGP-based Provider routers).

  - If the path selected by the remote PE router fails:

    1. The Provider routers switch the traffic destined for the multihomed CE device to the alternate path as soon as failure is detected.

    2. The Provider routers notify the remote PE routers of the path failure.

    3. The remote PE routers update their routing tables accordingly.

  For more information about the VPLS designated forwarder path selection algorithm, see "VPLS Path Selection Process for PE Routers" on page 753. This algorithm is also described in the Internet draft draft-kompella-l2vpn-vpls-multihoming-03.txt, *Multi-homing in BGP-based Virtual Private LAN Service*.

To enable the BGP path selection algorithm for Layer 2 VPN and VPLS routing instances, complete the following steps:

1. Run Junos OS on all of the PE and Provider routers participating in Layer 2 VPN or VPLS routing instances.

   Attempting to enable this functionality on a network with a mix of routers that both do and do not support this feature can result in anomalous behavior.

2. Specify a unique route distinguisher on each PE router participating in a Layer 2 VPN or VPLS routing instance.

3. Configure the `l2vpn-use-bgp-rules` statement on all of the PE and Provider routers participating in Layer 2 VPN or VPLS routing instances.

   You can configure this statement at the `[edit protocols bgp path-selection]` hierarchy level to apply this behavior to all of the routing instances on the router.

RELATED DOCUMENTATION

Understanding BGP Path Selection

VPLS Path Selection Process for PE Routers | **753**

path-selection

route-distinguisher

CHAPTER 13

Creating Backup Connections with Redundant Pseudowires

IN THIS CHAPTER

- Redundant Pseudowires for Layer 2 Circuits and VPLS | **224**
- Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | **226**

Redundant Pseudowires for Layer 2 Circuits and VPLS

IN THIS SECTION

- Types of Redundant Pseudowire Configurations | **225**
- Pseudowire Failure Detection | **225**

A redundant pseudowire can act as a backup connection between PE routers and CE devices, maintaining Layer 2 circuit and VPLS services after certain types of failures. This feature can help improve the reliability of certain types of networks (metro for example) where a single point of failure could interrupt service for multiple customers. Redundant pseudowires cannot reduce traffic loss to zero. However, they provide a way to gracefully recover from pseudowire failures in such a way that service can be restarted within a known time limit.

When you configure redundant pseudowires to remote PE routers, you configure one to act as the primary pseudowire over which customer traffic is being transmitted and you configure another pseudowire to act as a backup in the event the primary fails. You configure the two pseudowires statically. A separate label is allocated for the primary and backup neighbors.

For information about how to configure redundant pseudowires, see "Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS" on page 226.

The following sections provide an overview of redundant pseudowires for Layer 2 circuits and VPLS:

Types of Redundant Pseudowire Configurations

You can configure redundant pseudowires for Layer 2 circuits and VPLS in either of the following manners:

- You can configure a single active pseudowire. The PE router configured as the primary neighbor is given preference and this connection is the one used for customer traffic. For the LDP signalling, labels are exchanged for both incoming and outgoing traffic with the primary neighbor. The LDP label advertisement is accepted from the backup neighbor, but no label advertisement is forwarded to it, leaving the pseudowire in an incomplete state. The pseudowire to the backup neighbor is completed only when the primary neighbor fails. The decision to switch between the two pseudowires is made by the device configured with the redundant pseudowires. The primary remote PE router is unaware of the redundant configuration, ensuring that traffic is always switched using just the active pseudowire.

- Alternatively, you can configure two active pseudowires, one to each of the PE routers. Using this approach, control plane signalling is completed and active pseudowires are established with both the primary and backup neighbors. However, the data plane forwarding is done only over a one of the pseudowires (designated as the active pseudowire by the local device). The other pseudowire is on standby. The active pseudowire is preferably established with the primary neighbor and can switch to the backup pseudowire if the primary fails.

  The decision to switch between the active and standby pseudowires is controlled by the local device. The remote PE routers are unaware of the redundant connection, and so both remote PE routers send traffic to the local device. The local device only accepts traffic from the active pseudowire and drops the traffic from the standby. In addition, the local device only sends traffic to the active pseudowire. If the active pseudowire fails, traffic is immediately switched to the standby pseudowire.

The two configurations available for pseudowire redundancy have the following limitations:

- For the single active pseudowire configuration, it takes more time (compared to the two active pseudowire configuration) to switchover to the backup pseudowire when a failure is detected. This approach requires additional control plane signalling to complete the pseudowire with the backup neighbor and traffic can be lost during the switchover from primary to backup.

- If you configure two active pseudowires, bandwidth is lost on the link carrying the backup pseudowire between the remote PE router and the local device. Traffic is always duplicated over both the active and standby pseudowires. The single active pseudowire configuration does not waste bandwidth in this fashion.

Pseudowire Failure Detection

The following events are used to detect a failure (control and data plane) of the pseudowire configured between a local device and a remote PE router and initiates the switch to a redundant pseudowire:

- Manual switchover (user initiated)
- Remote PE router withdraws the label advertisement
- LSP to the remote PE router goes down
- LDP session with the remote PE router goes down
- Local configuration changes
- Periodic pseudowire OAM procedure fails (Layer 2 circuit-based MPLS ping to the PE router fails)

  When you configure a redundant pseudowire between a CE device and a PE router, a periodic (once a minute) ping packet is forwarded through the active pseudowire to verify data plane connectivity. If the ping fails, traffic is automatically switched to the redundant pseudowire.

When a failure is detected, traffic is switched from the failed active pseudowire to the redundant pseudowire. The redundant pseudowire is then designated as the active pseudowire. The switch is nonreversible, meaning that once the redundant pseudowire assumes the role of the active pseudowire at the time of a failover, it remains as the active pseudowire even though the previously active pseudowire comes up again.

For example, a primary pseudowire has failed and traffic has been successfully switched to the redundant pseudowire. After a period of time, the cause of the failure of the primary pseudowire has been resolved and it is now possible to reestablish the original connection. However, traffic is not switched back to the original pseudowire unless a failure is detected on the currently active pseudowire.

RELATED DOCUMENTATION

Example: Configuring H-VPLS Without VLANs | **860**

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS

IN THIS SECTION

- Configuring Pseudowire Redundancy on the PE Router | 227
- Configuring the Switchover Delay for the Pseudowires | 228
- Configuring a Revert Time for the Redundant Pseudowire | 228

A redundant pseudowire can act as a backup connection between PE routers and CE devices, maintaining Layer 2 circuit and VPLS services after certain types of failures. This feature can help improve the reliability of certain types of networks (metro for example) where a single point of failure could interrupt service for multiple customers. Redundant pseudowires cannot reduce traffic loss to zero. However, they provide a way to gracefully recover from pseudowire failures in such a way that service can be restarted within a known time limit.

For an overview of how redundant pseudowires work, see "Redundant Pseudowires for Layer 2 Circuits and VPLS" on page 224.

To configure pseudowire redundancy for Layer 2 circuits and VPLS, complete the procedures in the following sections:

Configuring Pseudowire Redundancy on the PE Router

You configure pseudowire redundancy on the PE router acting as the egress for the primary and standby pseudowires using the `backup-neighbor` statement.

To configure pseudowire redundancy on the PE router, include the `backup-neighbor` statement:

```
backup-neighbor {
    community name;
    psn-tunnel-endpoint address;
    standby;
    virtual-circuit-id number;
}
```

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

The `backup-neighbor` statement includes the following configuration options:

- `community`—Specifies the community for the backup neighbor.
- `psn-tunnel-endpoint`—Specifies the endpoint address for the packet switched network (PSN) tunnel on the remote PE router. The PSN tunnel endpoint address is the destination address for the LSP on the remote PE router.
- `standby`—Configures the pseudowire to the specified backup neighbor as the standby. When you configure this statement, traffic flows over both the active and standby pseudowires to the CE device. The CE device drops the traffic from the standby pseudowire, unless the active pseudowire fails. If the active pseudowire fails, the CE device automatically switches to the standby pseudowire.
- `virtual-circuit-id`—Uniquely identifies the primary and standby Layer 2 circuits. This option is configurable for Layer 2 circuits only.

Configuring the Switchover Delay for the Pseudowires

To configure the time the router waits before switching traffic from the failed primary pseudowire to a backup pseudowire, include the `switchover-delay` statement:

```
switchover-delay milliseconds;
```

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

Configuring a Revert Time for the Redundant Pseudowire

You can specify a revert time for redundant Layer 2 circuit and VPLS pseudowires. When you have configured redundant pseudowires for Layer 2 circuits or VPLS, traffic is switched to the backup pseudowire in the event that the primary pseudowire fails. If you configure a revert time, when the configured time expires traffic is reverted back to the primary pseudowire, assuming the primary pseudowire has been restored.

To configure a revert time for redundant pseudowires, specify the time in seconds using the `revert-time` statement:

```
revert-time (Protocols Layer 2 Circuits) seconds maximum seconds;
```

With the `maximum` option, specify a maximum reversion interval to add after the `revert-time` delay. If a revert-time delay is defined but a maximum timer is not defined, VCs are restored upon the revert-timer's expiration.

To reduce as much as possible the amount of traffic discarded, and potential data-path asymmetries observed during primary-to-backup transition periods, you can use this restoration timer. This restoration timer is activated when the backup path is performing as active, and then the primary path is restored. The goal is to avoid moving traffic back to the primary path right away, to make sure that the control plane's related tasks (such as IGP, LDP, RSVP, and internal BGP) have enough time to complete their updating cycle.

By enabling a gradual return of traffic to the primary path, you can ensure that the relatively-slow control-plane processing and updating does not have a negative impact on the restoration process.

The `maximum` option extends the revert timer’s functionality to provide a jittered interval over which a certain number of circuits can be transitioned back to the primary path. By making use of this maximum value, you can define a time interval during which circuits are expected to switch over. As a consequence, circuits’ effective transitions are scattered during restoration periods.

When making use of `revert-time` *`x`* `maximum` *`y`* statement, you can ensure that the corresponding circuit that is active is moved to the primary path within a time-slot (t1) such as that: x <= t1 <= y. In other words, by activating this statement, you can ensure the following:

- VCs stay in the backup path for at least x seconds after the primary path comes back up.
- VCs are moved back to the primary path before y seconds have elapsed.
- y maximum value = x maximum value * 2 = 1200 seconds.

The ideal values for x and y will are conditioned to internal aspects of your network. For this reason, there are no default values for these settings. If no revert-time is set, the default behavior is non-revertive. That is, circuits are not returned to the primary path upon restoration. They are kept on the backup path.

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

RELATED DOCUMENTATION

Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario | **388**

Example: Configuring H-VPLS Without VLANs | **860**

CHAPTER 14

Configuring Class of Service for Layer 2 VPNs

IN THIS CHAPTER

- Configuring Traffic Policing in Layer 2 VPNs | 230

Configuring Traffic Policing in Layer 2 VPNs

You can use policing to control the amount of traffic flowing over the interfaces servicing a Layer 2 VPN. If policing is disabled on an interface, all the available bandwidth on a Layer 2 VPN tunnel can be used by a single CCC or TCC interface.

For information on how to configure traffic policers, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

To enable Layer 2 VPN policing on an interface, include the `policer` statement:

```
policer {
    input policer-template-name;
    output policer-template-name;
}
```

If you configure CCC encapsulation, you can include the `policer` statement at the following hierarchy levels:

- `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family ccc]`
- `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family ccc]`

If you configure TCC encapsulation, you can include the `policer` statement at the following hierarchy levels:

- `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family tcc]`
- `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family tcc]`

For information about how to configure the encapsulation type, see "Configuring the Encapsulation Type" on page 154.

CHAPTER 15

Monitoring Layer 2 VPNs

IN THIS CHAPTER

- Configuring BFD for Layer 2 VPN and VPLS | **232**
- BFD Support for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS | **234**
- Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS | **235**
- Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview | **237**
- Configure a MEP to Generate and Respond to CFM Protocol Messages | **239**

Configuring BFD for Layer 2 VPN and VPLS

The following procedure describes how to configure Bidirectional Forwarding Detection (BFD) for Layer 2 VPN and VPLS. For VPNs, you configure the BFD sessions on the interfaces carrying traffic from the PE routers to the CE routers.

The BFD protocol is a simple hello mechanism that detects failures in a network. Hello packets are sent at a specified, regular interval. A neighbor failure is detected when the routing device stops receiving a reply after a specified interval. BFD works with a wide variety of network environments and topologies. The failure detection timers for BFD have shorter time limits than default failure detection mechanisms for BGP, so they provide faster detection.

The BFD failure detection timers are adaptive and can be adjusted to be faster or slower. The lower the BFD failure detection timer value, the faster the failure detection and vice versa. For example, the timers can adapt to a higher value if the adjacency fails (that is, the timer detects failures more slowly). Or a neighbor can negotiate a higher value for a timer than the configured value. The timers adapt to a higher value when a BFD session flap occurs more than three times in a span of 15 seconds. A back-off algorithm increases the receive interval by two if the local BFD instance is the reason for the session flap. The transmission interval is increased by two if the remote BFD instance is the reason for the session flap. You can use the `clear bfd adaptation` command to return BFD interval timers to their configured values. The `clear bfd adaptation` command is hitless, meaning that the command does not affect traffic flow on the routing device.

1. You can enable BFD failure detection. The BFD failure detection timers are adaptive and can be adjusted to be faster or slower. The lower the BFD failure detection timer value, the faster the failure

detection and vice versa. For example, the timers can adapt to a higher value if the adjacency fails (that is, the timer detects failures more slowly). Or a neighbor can negotiate a higher value for a timer than the configured value. The timers adapt to a higher value when a BFD session flap occurs more than three times in a span of 15 seconds. A back-off algorithm increases the receive (Rx) interval by two if the local BFD instance is the reason for the session flap. The transmission (Tx) interval is increased by two if the remote BFD instance is the reason for the session flap.

To enable BFD failure detection and specify the threshold for the adaptation of the BFD session detection time, specify a time in milliseconds using the threshold statement. When the detection time adapts to a value equal to or greater than the threshold, a single trap and a single system log message are sent.

NOTE: The threshold time must be equal to or greater than the value specified in the minimum-interval or the minimum-receive-interval statement.

You can use the `clear bfd adaptation` command to return BFD interval timers to their configured values. The `clear bfd adaptation` command is hitless, meaning that the command does not affect traffic flow on the routing device.

2. You can specify the minimum interval after which the local routing device transmits hello packets and then expects to receive a reply from a neighbor with which it has established a BFD session. You specify the interval in milliseconds using the *minimum-interval* statement.

   Optionally, instead of using this statement, you can specify the minimum transmit and receive intervals separately using the minimum-interval (specified under the `transmit-interval` statement) and minimum-receive-interval statements.

3. You can configure the minimum interval after which the local routing device must receive a reply from a neighbor with which it has established a BFD session. Specify the number of milliseconds using the *minimum-receive-interval* statement.

4. You can specify that an interface be declared down when a certain number of hello packets have not been received from a neighboring router through that interface. Specify the number of hello packets by including the *multiplier* statement.

5. You can configure BFD sessions not to adapt to changing network conditions by including the *no-adaptation* statement. We recommend that you *do not* disable BFD adaptation unless it is preferable to have BFD adaptation disabled in your network.

6. Specify the transmit interval options for `bfd-liveness-detection` statement by including the *transmit-interval* statement. The negotiated transmit interval for a peer is the interval between the sending of BFD packets to peers. The receive interval for a peer is the minimum time that it requires between packets sent from its peer; the receive interval is not negotiated between peers. To determine the transmit interval, each peer compares its configured minimum transmit interval with its peer's minimum receive interval. The larger of the two numbers is accepted as the transmit interval for that peer.

The `transmit-interval` statement specifies how often BFD statements are transmitted and includes the following options:

- `minimum-interval` *`milliseconds`*—Specify the minimum interval in milliseconds at which the local routing device transmits hello packets to a neighbor with which it has established a BFD session.

- `threshold` *`milliseconds`*—Specify the threshold for the adaptation of the BFD session transmit interval. When the transmit interval adapts to a value greater than the threshold, a single trap and a single system message are sent.

  NOTE: The threshold value specified in the `threshold` statement must be greater than the value specified in the `minimum-interval` statement for the `transmit-interval` statement.

7. Specify the BFD version by including the *version* statement. You can set BFD to version 1 or allow BFD to determine what version it needs to be by including the `automatic` option.

RELATED DOCUMENTATION

bfd-liveness-detection

clear bfd adaptation

BFD Support for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS

Bidirectional Forwarding Detection (BFD) support for virtual circuit connectivity verification (VCCV) enables you to configure a control channel for a pseudowire, in addition to the corresponding operations, administration, and management functions to be used over that control channel. For platform and Junos OS support information for BFD for VCCV or any other Junos OS feature, use Feature Explorer.

BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures. This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. Alternatively, you can use a ping operation to detect pseudowire failures. However, the processing resources required for a ping operation are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than a VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based), Layer 2 VPNs (BGP-based), and VPLS (LDP-based or BGP-based).

Junos OS has a distributed model for the BFD for VCCV. In legacy Junos OS releases the BFD for VCCV followed a Routing Engine-based implementation. In current Junos OS releases, the BFD for VCCV follows a distributed implementation over PIC concentrators, such as DPC, FPC, and MPC.

For distributed BFD, you need to configure the lo0 interface with unit 0 and the appropriate family enabled.

NOTE: For the distributed BFD for VCCV to work, you must configure MPLS family (`family mpls`) on the loopback interface.

```
user@router# set interfaces lo0 unit 0 family mpls
```

NOTE: On ACX Series routers, to enable BFD sessions over VCCV, you need configure control word. For VPLS, include the `control-word` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level. For Layer 2 VPN, include the `control-word` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols l2vpn]` hierarchy level. For Layer 2 Circuit, include the `control-word` statement at the `[edit protocols l2circuit neighbor` *`neighbor-ip-address`* `interface` *`interface-name`*`]` hierarchy level. The `control-word` statement must also be configured at the peer device for control word negotiation to happen between the pseudowire peers.

The periodic packet management process (ppmd) on the PIC concentrators handles the periodic packet management (send and receive) for BFD for VCCV. This enables Junos OS to create more BFD for VCCV sessions, and to reduce the time taken for error detection. Similarly, the distributed implementation improves the performance of Routing Engines because the Routing Engine resources used for BFD for VCCV implementation become available for Routing Engine-related applications when the BFD for VCCV-related processing moves to the PIC concentrators. The distributed BFD for VCCV implementation also enables the BFD for VCCV sessions to remain across graceful restarts.

RELATED DOCUMENTATION

Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS | **235**

Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS

Bidirectional Forwarding Detection (BFD) support for virtual circuit connection verification (VCCV) allows you to configure a control channel for a pseudowire, in addition to the corresponding operations and management functions to be used over that control channel. BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures.

This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. You can also use a ping operation to detect pseudowire failures. However, the processing resources required for a ping operation are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based), Layer 2 VPNs (BGP-based), and VPLS (LDP-based or BGP-based).

To configure OAM and BFD for Layer 2 VPNs, include the `oam` statement and sub-statements at the `[edit routing-instances *routing-instance-name* protocols l2vpn]` hierarchy level:

```
oam {
    bfd-liveness-detection;
    ping-interval ;
    ping-multiplier;
}
```

You can configure many of the same OAM statements for VPLS and Layer 2 circuits:

- To enable OAM for VPLS, configure the `oam` statement and substatements at the `[edit routing-instances *routing-instance-name* protocols vpls]` hierarchy level and at the `[edit routing-instances *routing-instance-name* protocols vpls neighbor *address*]` hierarchy level. The `pwe3-control-word` statement configured at the `[edit routing-instances *routing-instance-name* protocols l2vpn oam control-channel]` hierarchy level is not applicable to VPLS configurations.

- To enable OAM for Layer 2 circuits, configure the `oam` statement and substatements at the `[edit protocols l2circuit neighbor *address* interface *interface-name*]` hierarchy level. The `control-channel` statement and sub-statements configured at the `[edit routing-instances routing-instance-name protocols l2vpn oam]` hierarchy level do not apply to Layer 2 circuit configurations.

You can use the `show ldp database extensive` command to display information about the VCCV control channel and the `show bfd session extensive` command to display information about BFD for Layer 2 VPNs, Layer 2 circuits, and VPLS.

RELATED DOCUMENTATION

Junos OS Routing Protocols Library

Connectivity Fault Management Support for EVPN and Layer 2 VPN Overview

IN THIS SECTION

- LImitations of CFM on layer 2 VPN and EVPNs | **238**

The IEEE 802.1ag specification provides for Ethernet connectivity fault management (CFM). The goal of CFM is to monitor an Ethernet network that consists of one or more service instances through the use of CFM protocol messages. CFM partitions the service network into various administrative domains. Each administrative domain is mapped into a maintenance domain. A maintenance association end point (MEP) refers to the boundary of a domain. A MEP generates and responds to CFM protocol messages. You can configure multiple up (PE to PE) MEPs or down (PE to CE) MEPs for a single instance of a maintenance domain identifier and a maintenance association name to monitor services in a VPN.

For Layer 2 VPNs and EVPN networks, you can configure multiple up MEPs for a single combination of maintenance association ID and maintenance domain ID on a routing instance on the logical interfaces (IFLs), regardless of whether the logical interface is composed of physical interfaces on the same device or on different devices. The devices must be in enhanced IP network services mode.

In an EVPN network, the following CFM features are supported:

- Monitoring the connectivity between two provider edge (PE) routers in an active-active or active-standby multihomed configuration.

- Delay measurement and Synthetic Loss measurement. This feature is not supported when multiple MEPs are configured on multiple logical interfaces (IFLs) on the same physical interface (IFD).

- CFM monitoring between PE devices and customer edge (CE) devices. When the customer edge device is not a Juniper Networks device, you can enable CFM monitoring by using either the remote defect indication (RDI) bit or the Interface Status TLV. For more information, see Understanding CFM Monitoring between CE and PE Devices

- Starting with 18.3R1, Junos OS supports CFM configuration in attachment circuits (AC) on EVPN with ETREE services. The AC is a physical or virtual circuit that connects a CE device to a PE device. You can configure multiple Up MEPs on the MD or MA to monitor each AC between the CE and PE.

- Starting with 18.3R1, Junos OS supports maintenance intermediate points (MIPs) on Attachment Circuits on EVPN with ETREE and EVPN with ELAN services. When you configure MIP using the named bridge domain, all the interfaces will be enabled except for the EVPN core interface. For more information on MIPS, see Configuring Maintenance Intermediate Points (MIPs).

- Starting with 19.2R1, Junos OS supports MEPs and MIPs on ACs in an EVPN-VPWS network. CFM monitoring on EVPN-VPWS supports continuity check messages (CCM), delay measurements, synthetic loss measurement, loopback and link trace messages on single-active multihomed networks.

- Starting in Junos OS Release 20.2R1, Junos OS supports inline performance monitoring on an EVPN network. When inline performance monitoring is enabled, the router offloads performance monitoring packet processing from the CPU to other hardware, thereby decreasing the load on the CPU and allowing an increase in the number of performance monitoring session. Inline performance monitor only applies to delay measurements (DM) and synthetic loss measurements (SLM).

  For more information on inline performance monitoring, see Enabling Inline Mode Of Performance Monitoring To Achieve Maximum Scaling.

  For more information on configuring delay measurment, see Configuring Ethernet Frame Delay Measurement Sessions.

  For more information on configuring synthetic loss measurement, see Configuring Ethernet Synthetic Loss Measurements.

LImitations of CFM on layer 2 VPN and EVPNs

- In a circuit cross-connect (ccc) layer 2 VPN or local switch with MEPs and maintenance intermediate points (MIPs), the counter for link trace messages (LTMs) received on the MAC address of the up MEP does not get incremented when the MIP in the path is configured at the same level. The MIP traps the LTM packet, while the LTR message is sent. This leads to a discrepancy between the number of LTMs received and the number of LTRs sent.

- CFM up MEP on an EVPN network does not support the use of action profiles for interface down. In other words, you can configure an action profile, but no action is taken.

- CFM up MEP is supported on EVPN with ELAN and EVPN with ETREE services.

- CFM monitoring between leaf nodes on EVPN with ETREE services is not supported. CFM monitors MEP session from a leaf node to a root node and from a root node to another root node.

- CFM monitors the AC connectivity from between PE devices, learning about local adjacencies. In EVPN with E-TREE services, performance monitoring on local adjacencies is not supported.

For more information on CFM, see IEEE 802.1ag OAM Connectivity Fault Management Overview.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
19.2R1	Starting with 19.2R1, Junos OS supports MEPs and MIPs on ACs in an EVPN-VPWS network.
18.3R1	Starting with 18.3R1, Junos OS supports CFM configuration in attachment circuits (AC) on EVPN with ETREE services.
18.3R1	Starting with 18.3R1, Junos OS supports maintenance intermediate points (MIPs) on Attachment Circuits on EVPN with ETREE and EVPN with ELAN services.

RELATED DOCUMENTATION

IEEE 802.1ag OAM Connectivity Fault Management Overview

Creating a Maintenance Domain

Creating a Maintenance Association

Configuring a MEP to Generate and Respond to CFM Protocol Messages

show oam ethernet connectivity-fault-management interfaces

Configure a MEP to Generate and Respond to CFM Protocol Messages

IN THIS SECTION

- Configure a Maintenance Association End Point (MEP) | 240
- Configure a Remote Maintenance Association End Point (MEP) | 242

A maintenance association end point (MEP) refers to the boundary of a domain. A MEP generates and responds to connectivity fault management (CFM) protocol messages. You can configure multiple up MEPs for a single combination of maintenance association ID and maintenance domain ID for interfaces belonging to a particular VPLS service or a bridge domain. You can configure multiple down MEPs for a single instance of maintenance domain identifier and maintenance association name to monitor services provided by Virtual Private LAN service (VPLS), bridge domain, circuit cross-connect (CCC), or IPv4 domains.

For layer 2 VPNs routing instances (local switching) and EVPN routing instances, you can also configure multiple up MEPs for a single combination of maintenance association ID and maintenance domain ID on logical interfaces. The logical interface can be configured on different devices or on the same device. To support multiple up MEPs on two IFLs, enhanced IP network services must be configured for the chassis.

You can enable automatic discovery of a MEP. With automatic discovery a MEP is enabled to accept continuity check messages (CCMs) from all remote MEPs of the same maintenance association. If automatic discovery is not enabled, the remote MEPs must be configured. If the remote MEP is not configured, the CCMs from the remote MEP are treated as errors.

Continuity measurement is provided by an existing continuity check protocol. The continuity for every remote MEP is measured as the percentage of time that remote MEP was operationally up over the total administratively enabled time. Here, the operational uptime is the total time during which the CCM adjacency is active for a particular remote MEP and the administrative enabled time is the total time during which the local MEP is active. You can also restart the continuity measurement by clearing the currently measured operational uptime and the administrative enabled time.

Configure a Maintenance Association End Point (MEP)

To configure a maintenance association end point:

1. Specify an ID for the MEP at the [`edit protocols oam ethernet connectivity-fault-management maintenance-domain` *`domain-name`* `maintenance-association` *`ma-name`*]. You can specify any value from 1 through 8191.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name]
user@host# set mep mep-id
```

2. Enable maintenance end point automatic discovery so the MEP can accept continuity check messages (CCMs) from all remote MEPs of the same maintenance association.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name mep mep-id]
user@host# set auto-discovery
```

3. Specify the direction in which the CCM packets are transmitted for the MEP. You can specify up or down. If you specify the direction as up, CCMs are transmitted out of every logical interface that is part of the same bridging or VPLS instance except for the interface configured on the MEP. If you specify the direction as down, CCMs are transmitted only out of the interface configured on the MEP.

NOTE: Ports in the Spanning Tree Protocol (STP) blocking state do not block CFM packets destined to a down MEP. Ports in an STP blocking state without the continuity check protocol configured do block CFM packets.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name mep mep-id]
user@host# set direction down
```

4. Specify the interface to which the MEP is attached. It can be a physical interface, logical interface, or trunk interface. On MX Series routers, the MEP can be attached to a specific VLAN of a trunk interface.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name mep mep-id]
user@host# set interface interface-name
```

5. Specify the IEEE 802.1 priority bits that are used by continuity check and link trace messages. You can specify a value from through 7 as the priority.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name mep mep-id]
user@host# set priority number
```

6. Specify the lowest priority defect that generates a fault alarm whenever CFM detects a defect. Possible values include: all -defects, err-xcon, mac-rem-err-xcon, no-defect, rem-err-xcon, and xcon.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-name mep mep-id]
user@host# set lowest-priority-defect mac-rem-err-xcon
```

7. Specify the ID of the remote MEP at the `[edit protocols oam ethernet connectivity-fault-management maintenance-domain` *`domain-name`* `maintenance-association` *`ma-name`* `mep` *`mep-id`*`]`. You can specify any value from 1 through 8191.

```
[edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
maintenance-association ma-namemep mep-id]
user@host# set remote-mep mep-id
```

SEE ALSO

priority

Configure a Remote Maintenance Association End Point (MEP)

To configure a remote maintenance association end point:

1. Configure the remote MEP by specifying the MEP ID at the [`edit protocols oam ethernet connectivity-fault-management maintenance-domain` *`domain-name`* `maintenance-association` *`ma-name`* `mep` *`mep-id`*]. You can specify any value from 1 through 8191.

   ```
   [edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
   maintenance-association ma-namemep mep-id]
   user@host# edit remote-mep mep-id
   ```

2. Specify the name of the action profile to be used for the remote MEP by including the `action-profile` *`profile-name`* statement at the [`edit protocols oam ethernet connectivity-fault-management maintenance-domain` *`domain-name`* `maintenance-association` *`ma-name`* `mep` *`mep-id`* `remote-mep` *`remote-mep-id`*]. The profile must be defined at the [`edit protocols oam ethernet connectivity-fault-management`] hierarchy level.

   ```
   [edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
   maintenance-association ma-namemep mep-id remote-mep remote-mep-id]
   user@host# set action-profile profile-name
   ```

3. Configure the remote MEP to detect initial loss of connectivity. By default, the MEP does not generate loss-of-continuity (LOC) defect messages. When you configure the `detect-loc` statement, a loss-of-continuity (LOC) defect is detected if no continuity check message is received from the remote MEP within a period equal to 3.5 times the continuity check interval configured for the maintenance association. If a LOC defect is detected, a syslog error message is generated.

   NOTE: When you configure connectivity-fault management (CFM) along with `detect-loc`, any `action-profile` configured to bring down the interface is executed if continuity check message is not received . However, the `action-profile` is not executed if you have not configured `detect-loc` and continuity check message is not received.

   ```
   [edit protocols oam ethernet connectivity-fault-management maintenance-domain domain-name
   maintenance-association ma-namemep mep-id remote-mep remote-mep-id]
   user@host# set detect-loc
   ```

SEE ALSO

remote-mep

RELATED DOCUMENTATION

action-profile

auto-discovery

connectivity-fault-management

detect-loc

direction

lowest-priority-defect

3

PART

Configuring Group VPNs

- Configuring Group VPNv2 | **245**

CHAPTER 16

Configuring Group VPNv2

IN THIS CHAPTER

- Group VPNv2 Overview | **245**
- Configuring Group VPNs in Group VPNv2 on Routing Devices | **269**
- Group VPN on AMS interfaces | **272**
- Use Case for Configuring Group VPNv2 | **273**
- Example: Configuring Group VPNs in Group VPNv2 on Routing Devices | **274**

Group VPNv2 Overview

IN THIS SECTION

- Group VPNv2 Technology Overview | **245**
- Group VPNv2 Implementation Overview | **255**

Group VPNv2 Technology Overview

NOTE: Group VPNv2 is the name of the Group VPN technology on some router platforms. Group VPNv2 is different from the Group VPN technology implemented on SRX Security Gateways. The term Group VPN is sometimes used in this document to refer to the technology in general, not to the SRX technology.

For more information about Group VPN on SRX Security Gateway devices, see Group VPNv2 Overview.

For more information on platform and release support for any Junos OS feature, use Feature Explorer.

This section explains the technological concepts of Group VPNv2.

Understanding Group VPNv2

Group VPN is a trusted group to eliminate point-to-point tunnels and their associated overlay routing. All group members share a common security association (SA), known as a group SA (GSA). The GSA enables group members to decrypt traffic that was encrypted by any other group member. Starting in Junos OS Release 18.2R1, we confirm the Group VPN redundancy with service redundancy protocol [SRD] running on MX routers. MX routers with redundancy between them acts as Group VPN members. For more details on service redundancy protocol see Service Redundancy Daemon Overview.

Starting in Junos OS Release 15.1, Junos OS supports Group VPNv2. Group VPNv2 is a category of VPN that eliminates the need for point-to-point VPN tunnels in a mesh architecture. It is a set of features that are necessary to secure unicast traffic over a private WAN that originates on or flows through a router.

Group VPNv2 introduces the concept of a trusted group to eliminate point-to-point tunnels and their associated overlay routing. All group members share a common security association (SA), also known as a group SA. This enables group members to decrypt traffic that was encrypted by any other group member.

Group VPNv2 provides the following advantages:

- Provides data security and transport authentication, helping to meet security compliance and internal regulation by encrypting all WAN traffic.
- Enables high-scale network meshes and eliminates complex peer-to-peer key management with group encryption keys.
- Reduces the number of endpoint changes that need to be made due to group member change or policy change.
- Maintains network intelligence such as full-mesh connectivity, natural routing path, and quality of service (QoS) in MPLS networks.
- Grants authenticated membership control with a centralized key server.
- Allows encryption and decryption of traffic among all group members defined in the group policy.
- Helps to ensure low latency and jitter by enabling full-time, direct communications between sites, without requiring transport through a central hub.
- Reduces traffic loads on customer premises equipment (CPE) and provider edge (PE) encryption devices by using the core network for traffic replication, avoiding packet replication at each individual peer site.

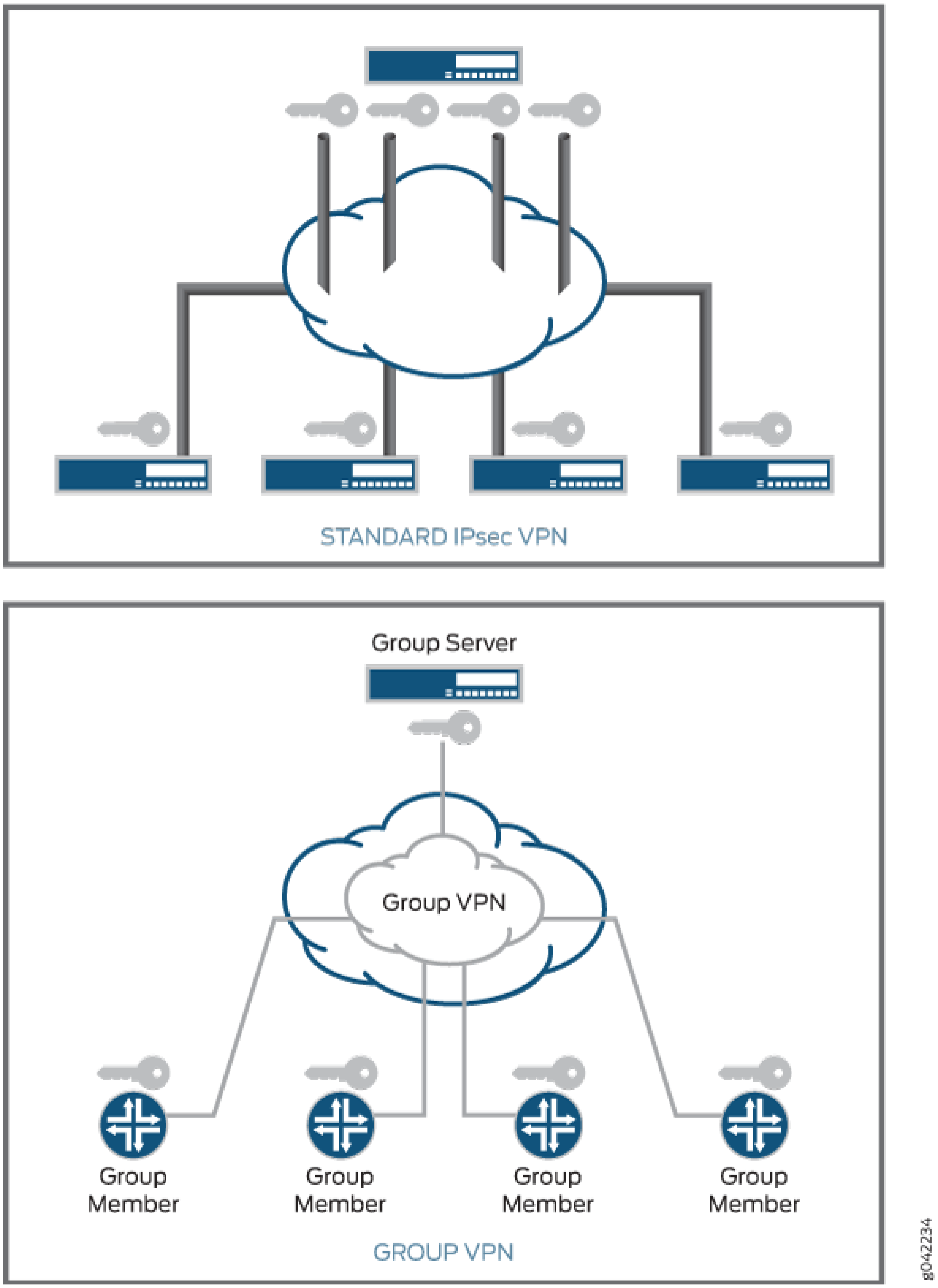
Group VPNv2 and Standard IPsec VPN

Group VPNv2 is built on standards-based technologies that integrate routing and encryption together in the network. An IPsec security SA is a unidirectional agreement between VPN participants that defines the rules to use for authentication and encryption algorithms, key exchange mechanisms, and secure communications.

Traditional IPsec VPN deployments tackle the problem of securing traffic between gateways in the network by creating an overlay network based on the use of point-to-point tunnels. Traffic carried over these tunnels is normally encrypted and authenticated in order to provide data integrity and confidentiality. Secure group members are managed through the Group Domain of Interpretation protocol (GDOI). The GDOI solution takes a different approach by disassociating the encryption and authentication problem from the transport. By doing this, GDOI-based solutions provide a way to encrypt branch-to-branch communications without the need to configure branch-to-branch tunnels.

With current VPN implementations, the SA is a point-to-point tunnel between two end points. Group VPNv2 extends the IPsec architecture to support SAs that are shared by a group of routers (see Figure 14 on page 248). A key server distributes keys and policies to all registered and authenticated member routers. By distributing policies from a centralized point and by sharing the same group security association (the entire group has a single Phase 2 SA) with authenticated group members, key distribution and management are greatly simplified.

Figure 14: Standard IPsec VPN and Group VPNv2



Group VPNv2 is a client/server architecture. All members have a unique Phase 1 IKE SA with the key server. Hence, if there are `n` members, there is a total of `n` Phase 1 IKE SAs. However, the entire group shares a single Phase 2 SA.

In traditional IPsec, the tunnel endpoint addresses are used as a new packet source and destination. The packet is then routed over the IP infrastructure, using the encrypting end point source IP address and the decrypting end point destination IP address. In the case of Group VPN, IPsec-protected data packets preserve the original source and destination addresses of the host in the outer IP header in order to preserve the IP address. This is known as tunnel header preservation. The biggest advantages of tunnel header preservation is the ability to route encrypted packets using the underlying network routing infrastructure.

Table 3: Group VPN vs Traditional Point-to-Point IPsec

Feature	Traditional Point-to-Point IPsec Tunnels	Group VPN
Scalability	IKE/IPsec tunnels between each pair of peers increases management and configuration complexity.	Single SA and key pair used for the entire any-to-any group. Reduced management and configuration complexity.
Any-to-any instant connectivity	Cannot be done to scale due to management and configuration complexity.	Scales well due to the use of GDOI and shared SA within the group.
Overlay routing	Requires overlay routing.	No overlays-native routing.
IP Header Preservation	New IP Header added to original packet results in limited advanced quality of service (QoS). Will work in NAT environments.	Keeps original IP header on IPsec packet. Preserves advanced QoS capabilities. Will not work in NAT environments.

Understanding the GDOI Protocol

The Group Domain of Interpretation (GDOI) protocol described in RFC 6407 is used to distribute a set of cryptographic keys and policies to a group of devices. GDOI is defined as the Internet Security Association Key Management Protocol (ISAKMP) Domain of Interpretation (DOI) for group key management. In a group management model, the GDOI protocol operates between a group member and a group controller or key server (GC/KS) and manages group security associations and group-keys for a set of security participants. The ISAKMP defines two phases of negotiation. GDOI is a Phase 2 protocol protected by a Phase 1 ISAKMP security association. IKEv1 is specified in RFC 6407 as a Phase 1 protocol.

GDOI introduces two different encryption keys:

- Key encryption key (KEK)—Used to secure the control plane. KEK is the name of the key used by the group members to decrypt rekey messages from the GC/KS. This key is part of the Security Association Key Encryption Key (SAK).
- Traffic encryption key (TEK)—Used to secure the data plane. TEK is the name of the key used by the group members to encrypt or decrypt communication between other group members. This key is part of the Security Association Transport Encryption Key (SA TEK).

As with standard IPsec, all keys have a lifetime and have to be rekeyed. The keys distributed through GDOI are group keys and are used by the entire group.

The group SAs and key management are handled through two types of GDOI exchanges:

- `groupkey-pull`—This exchange allows a member to request SAs and keys shared by the group from the server.

  In the pull method, the group member requests the group SA and policy from the key server. This request is protected over the IKE SA.

  The `groupkey-pull` is the first exchange in the GDOI protocol and is used for group member registration with the GC/KS. The group member specifies the group with which it wants to register, and the GC/KS sends all necessary group SAs and keys to the group member if the member is authorized to join the group. The complete exchange is secured by a Phase 1 SA (IKEv1 SA), which is established with IKEv1 before the `groupkey-pull` exchange begins. The `groupkey-pull` is part of Phase 2 of the GDOI protocol.
- `groupkey-push`—This exchange is a single rekey message that allows the server to send group SAs and keys to members before existing group SAs expire. Rekey messages are unsolicited messages sent from the server to members.

  The `groupkey-push` is the second exchange in the GDOI protocol and is initiated by the GC/KS to all registered members of the group. Table 4 on page 251 shows the payloads that the MX Series group member expects to receive in `groupkey-push` messages.

Table 4: groupkey-push Message Payloads

Payload	Description
group associated policy (GAP)	A GAP payload allows for the distribution of group-wide policy, such as instructions as to when to activate and deactivate SAs. This payload contains values for activation time delay (ATD) and deactivation time delay (DTD) for the traffic encryption key (TEK) as well as IP-Delivery Delayed Detection Protocol window type and window size for the IPsec traffic.
Security Association Transport Encryption Key (SA TEK)	Traffic selectors.
Security Association Key Encryption Key (SAK) (Optional)	The security association (SA) for the key encryption key (KEK). Also known as SA KEK. **NOTE**: `groupkey-push` messages that do not include the optional payloads are still valid messages.
traffic encryption key (TEK) (Optional)	Key for encrypting the data traffic between group members.
key encryption key (KEK) (Optional)	Used to protect the TEK.

The `groupkey-push` exchange is secured by an SA KEK (SAK), which is installed during the `groupkey-pull` exchange. The `groupkey-push` is part of Phase 2 of the GDOI protocol.

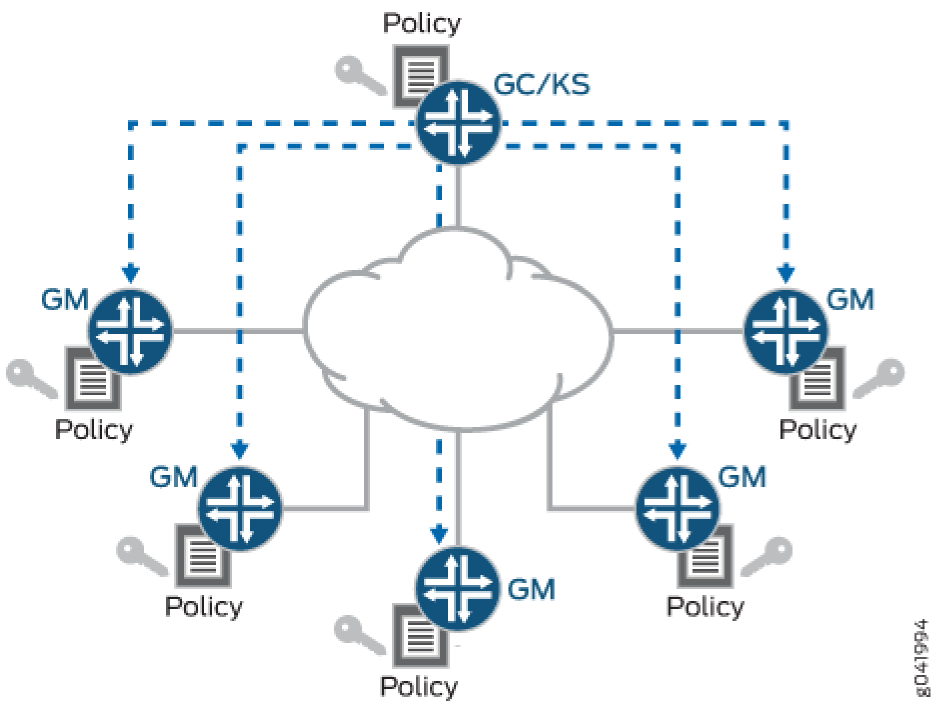
In some cases, the GC/KS may want to receive group key push acknowledgment messages from group members. The push acknowledgment messages from group members confirms that the member received the message and has taken action on their policy. The GC/KS can also use the acknowledgment to determine which group members are receiving the current group policy and which group members are no longer participating in the group. Starting in Junos OS 19.2R1, Junos OS sends an acknowledgment message with SHA-256 checksum when it receives a groupkey push message with a standard KEK_ACK_REQUESTED value of 9 in the SA KEK payload as defined in RFC 8263 or a KEK_ACK_REQUESTED value of 129 that is used is older key servers.

GDOI Protocol and Group VPNv2

Group VPNv2 is the name of the security technology implemented on the router platforms from Juniper Networks. Group VPNv2 uses the GDOI protocol (RFC 6407) as a base, in addition to other functionalities.

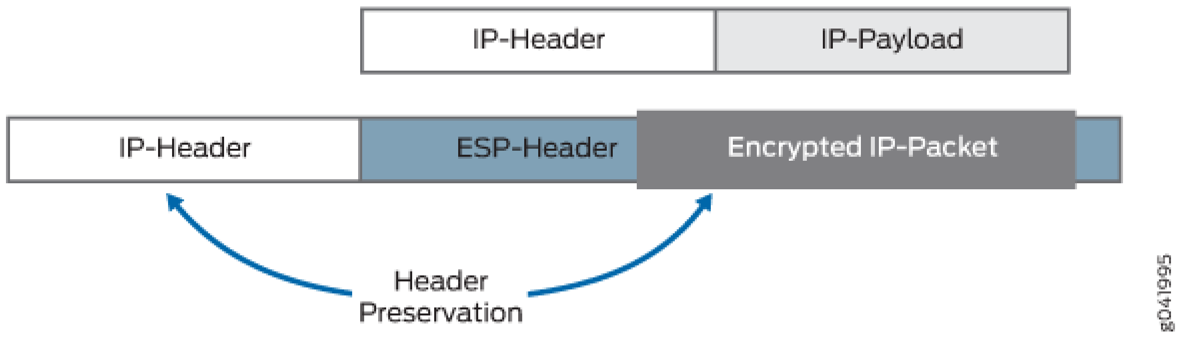
Group VPNv2 technology is based on the GDOI protocol to handle the most important functionality. This protocol is specified in RFC 6407 and defines an ISAKMP Domain of Interpretation (DOI) to manage group SAs and keys for a group of security participants. Thus, all members of the group share identical information to encrypt and decrypt traffic among each other. The creation, management, and distribution of group SAs and group keys are centralized and performed by the GC/KS. Figure 15 on page 252 provides a brief overview of the Group VPNv2 functionality using GDOI.

Figure 15: Group VPNv2 Using GDOI



The group members use the Encapsulating Security Payload (ESP) protocol in tunnel mode to secure the traffic. However, in Group VPN the tunnel mode is modified. Because there is no direct association between the group members, it is not necessary to use special IP addresses in the outer IP header (that is, IP addresses of IPsec gateways). Every group member can decrypt the traffic of every other group member. Thus, the inner IP-Header is copied to the outer IP-Header, and the underlying routing infrastructure and QoS infrastructure can be used. This feature is called Header Preservation and is shown in Figure 16 on page 253.

Figure 16: Header Preservation



To get group SAs and group keys, the group member must register with the GC/KS for a specific group. The result is an IKEv1 SA, which is only needed to secure the registration process. After the registration, the group member has all the information to communicate with the other group members (SA TEK), as well as the information to successfully decrypt the rekeying messages (SAK). The GC/KS sends out rekeying messages before either the SA TEK or SAK lifetime expires. It is also possible to send an SA TEK update as well as a SAK update in the same rekey message. The IKEv1 SA is no longer needed and is deleted after the lifetime expires (no IKEv1-rekeying).

Group VPNv2 Traffic

Group VPNv2 traffic includes:

- Control-plane-traffic—Traffic from the group members to the GC/KS in the Group VPNv2 deployment with the GDOI protocol only.
- Data-plane-traffic—Traffic between the group members in the Group VPNv2 deployment with the ESP protocol only which is already known from IPsec.

Group Security Association

Unlike traditional IPsec encryption solutions, Group VPN uses the concept of group security association. A group SA is similar to an SA in terms of functionality. Group SAs are shared among all group members of a common GDOI group. All members in the Group VPN group can communicate with each other using a common encryption policy and a shared group SA. With a common encryption policy and a shared group SA, there is no need to negotiate IPsec between group members. This reduces the resource load on the group members. Traditional IPsec scalability issues (number of tunnels and associated SAs) do not apply to Group VPN group members.

Group Controller/Key Server

A group controller or key server (GC/KS) is a device used for creating and maintaining the Group VPNv2 control plane. It is responsible for creation and distribution of group SAs and group keys. All information the group members need to communicate with other group members is provided by the GC/KS. All encryption policies, such as interesting traffic, encryption protocols, security association, rekey timers, and so on, are centrally defined on the GC/KS and are pushed down to all group members at registration time. Group members authenticate with the GC/KS using IKE Phase 1 and then download the encryption policies and keys required for Group VPN operation. The GC/KS is also responsible for refreshing and distributing the keys.

NOTE: The GC/KS functionality is not supported on MX Series routers. The MX Series routers that are configured as group members can connect with Cisco GC/KS only. There is no support for MX Series group members to interact with the Juniper Networks SRX Series acting as a GC. See Table 7 on page 267 for compatibility between the various types of group members and GC/KSs.

Group Member

A group member is an IPsec endpoint device used for the traffic encryption process and is responsible for the actual encryption and decryption of data traffic. A group member is configured with IKE Phase 1 parameters and GC/KS information. Encryption policies are defined centrally on the GC/KS and downloaded to the group member at the time of successful registration. Each group member then determines whether incoming and outgoing traffic should be decrypted or encrypted (using its SA) based on its group membership.

From a functionality point of view, a group member is similar to an IPsec gateway. However, the SAs in normal IPsec exist between two IPsec gateways. In GDOI, the group member registers with the GC/KS in order to participate in the Group VPN. During registration, the group member provides the group ID to the GC/KS to get the respective policies, SAs, and keys needed for this group. Rekeying is accomplished by the group members through the `groupkey-pull` method (re-registration) or by the GC/KS through the `groupkey-push` method.

Anti-Replay Protection for Group VPNv2 Traffic

Since Group VPN communication is essentially any-to-any communication over the same shared security association, the use of sequence numbers for anti-replay protection does not work. Because of this, Junos OS supports an IETF draft specification for a time-based anti-replay mechanism, draft-weis-delay-detection-01. It is available at http://tools.ietf.org/html/draft-weis-delay-detection-01 .

To implement this feature, MX Series member routers make use of a new IP Delivery Delay Detection Protocol time-stamp header within the packet. See "Implementing IP Delivery Delay Detection Protocol (Time-Based Anti-Replay Protection)" on page 262 for details.

Partial Fail-Open on MX Series Member Routers

Group members in a Group VPN rely on the GC/KS to generate keying material for the shared SA. Therefore, connectivity between the group members and GC/KSs is required to initially protect traffic and to continually protect traffic over rekey events. In the event of communication failure between the group member and the GC/KS, the default behavior of the group members is to stop forwarding traffic. This is known as fail-closed.

A nondefault configuration option is available to permit some specifically defined traffic to flow through the group member without being encrypted until such time as the member is able to contact the GC/KS and retrieve the active SA. This is known as partial fail-open.

The partial fail-open feature requires a policy configuration option that creates a rule on the applicable MX Series group member for a particular Group VPNv2 defined by source and destination addresses. This fail-open rule is active only when group SA is in disabled state because of connectivity failure with the key server. Traffic that would normally pass through the Group VPN but does not match the fail-open rule is dropped. More than one fail-open rule can be defined for the Group VPN object. If no fail-open rules are configured, then the fail-open feature is disabled.

Group VPNv2 Implementation Overview

This section explains the Juniper Networks solution for implementing Group VPNv2.

Enabling Group VPNv2

A service set is used to enable Group VPNv2 on a particular interface on applicable MX Series routers.

Configuring the Service Set

Group VPNv2 is configured inside a service set using the `ipsec-group-vpn` statement at the `[edit services service-set *service-set-name*]` hierarchy level.

Sample Service Set Configuration

```
[edit services]
service-set service-set-name {
    interface-service {
        service-interface service-interface-name;
    }
}
ipsec-group-vpn vpn-name;
```

NOTE:

- Only one group member can be configured per service set.
- Next-hop style service set is not supported with Group VPNv2.

Applying the Service Set

A service set is applied at the interface level.

Sample Applying Service Set Configuration

```
[edit interfaces]
interface-name {
    unit 0 {
        family inet {
            service {
                input {
                    service-set service-set-name;
                }
                output {
                    service-set service-set-name;
                }
            }
            address 10.0.30.2/30;
        }
    }
}
```

Packet Steering

The interface-style service set configuration is used to steer traffic from the Packet Forwarding Engine to the PIC. Packets received on an interface with a service set pointing to the Group VPNv2 object are forwarded to the PIC by being injected into the corresponding service interface.

Registering a Group Member

The group member registration to the server starts when the `ipsec-group-vpn` statement is configured for a service set and the service interface is up. When the service interface goes down, all group SAs associated with this interface are cleared, and no registration is triggered for these Group VPNs until the interface comes up.

Group member registration involves establishing IKE SA with the GC/KS followed by a `groupkey-pull` exchange to download the SAs and the traffic keys for the specified group identifier.

NOTE: Junos OS does not support traffic-based SA negotiation triggering for Group VPNs in Group VPNv2.

Rekeying a Group Member (groupkey-push Method)

The GC/KS will send out a unicast `groupkey-push` message to registered group members in order to:

- Send new key encryption keys (KEKs) or traffic encryption keys (TEKs).

  The push messages can contain all or only some of the payload elements shown in Table 4 on page 251. When the GAP payload contains both old SAs and new replacement SAs, the group member router will apply the ATD and DTD values as a normal rekey by means of push. If there is no ATD value in the update, the member router installs the new SAs immediately. If there is no DTD value, the old SAs will remain in place until their expiration.

- Update group associated policy (GAP) for an existing SA.

  A GC/KS can send a unicast push message to update the configuration to group members at any time. The GAP payload can include configuration changes to the IP Delivery Delay Detection Protocol, encryption algorithm, lifetime, and so on. The updated configuration is either applied immediately or with a delay. ATD and DTD values are used to achieve the timing for the activation of the new TEK and deletion of the existing TEK, respectively. If the existing TEK lifetime has to be reduced, then the DTD value is set accordingly in the push message. The new TEK in the push message is activated based on the ATD value in the payload.

- Send delete key notifications for TEK or KEK.

The GC/KS can send the optional delete notification payload in the push message for deleting keys and SAs on the member. The push message contains the protocol ID that indicates whether the delete notification is for TEK or KEK. The group member router deletes the key based on the group ID and SPI value contained in the payload. Deleting a specific TEK or KEK can be done with a delay value specified in the DTD attribute. If the delay value is 0, and the payload contains a specific SPI, then the matching TEK or KEK is deleted immediately. If all the TEKs or KEKs (or both) need to be deleted in the group, then the SPI value is set to 0 for the corresponding protocol ID in the payload.

- Remove a member router from the Group VPN in Group VPNv2.

  The push messages are used to allow the GC/KS to delete members from the Group VPN. In one case, the GC/KS sends a rekey message with only the old SAs and a smaller DTD value. The group member router installs the new, smaller DTD value. Since it did not receive new SA keys, the member router tries to re-register using the `groupkey-pull` method. This re-registration attempt is rejected by the GC/KS, thus deleting the member from the Group VPN. In the second case, the GC/KS sends a delete payload for the SPI of the old SA. The group member router deletes the SA immediately and attempts to re-register using the `groupkey-pull` method. This re-registration attempt is rejected by the GC/KS, thus deleting the member from the Group VPN.

Registered MX Series group members send a unicast PUSH ACK message back to the GC/KS to acknowledge the receipt of the original push message.

Rekeying a Group Member (groupkey-pull Method)

For group member rekeying, using the `groupkey-pull` method, the group members typically re-register with the GC/KS when there is between 7 percent and 5 percent remaining in the existing TEK or KEK soft lifetime. If the existing IKE SA is available, it is used in the pull message. After the GC/KS responds with a new key, both the old key and the new key can be used for decryption. However, the new key is not used for encryption until 30 seconds of lifetime of the old key is remaining. If the existing IKE SA is not available, the pull message results in new IKE negotiations between the group member and the GC/KS.

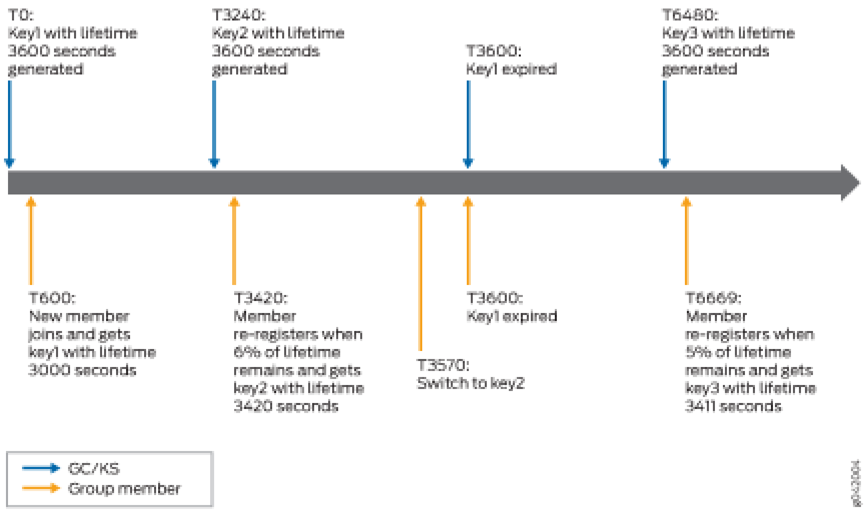
Upon receiving the pull message regarding a specific Group VPN from the group member, the GC/KS responds with all the TEKs and the KEK for that group.

If any existing SA is not included in the response from the GC/KS, then the missing SAs are deleted by the group member.

Taking as an example, the GC/KS is configured with a lifetime of 3600 seconds and is connected to one group member without retransmit. Based on the server configuration, the GC/KS generates a new key when 10 percent of the lifetime is remaining. The group member, however, re-registers with the GC/KS when 5 percent to 7 percent of the lifetime is remaining.

Figure 17 on page 259 represents the rekeying process between the GC/KS and the group member.

Figure 17: Group Member Rekeying



Authenticating a Group Member

Junos OS does not provide public key infrastructure (PKI) support for Group VPN in Group VPNv2. As a result, pre-shared keys are used for group member authentication.

Fragmenting Group VPNv2 Traffic

Because of the header preservation functionality and the usage of the underlying routing infrastructure, it is necessary to fragment the packets before encryption occurs (if it cannot be prevented).

Hence, pre-fragmentation is supported and is recommended for all deployments.

To avoid post-fragmentation, set the `clear`, `set`, and `copy` options for the DF bit in the Group VPNv2 configuration.

Based on this flag setting, the IPsec header has either the `df-bit` set to `clear`, `set`, or `copy` from the inner packet.

NOTE: The DF bit has the `clear` option set as default.

Sample DF Bit Configuration

```
[edit]
security {
    group-vpn {
        member {
            ipsec {
                vpn group-vpn-name {
                    df-bit clear;
                }
            }
        }
    }
}
```

Encrypting Group VPNv2 Traffic

Group members encrypt traffic based on the group SAs and keys provided by the GC/KS. The Group VPNv2 encryption path is as follows:

1. Packet received by the Packet Forwarding Engine is checked against a flow match. If a match is found, the packet is further processed and transmitted.

2. If a match is not found, a rule lookup is performed. If a match is found, a flow is created, and the packet is further processed and transmitted.

3. If the rule lookup fails, the packet is dropped.

NOTE: Group SA is not triggered during packet processing.

Decrypting Group VPNv2 Traffic

After registration is successful and Group VPN SAs are installed, an ESP session is created. Group VPNv2 creates the ESP session with a zero source and destination IP. Because the ESP session is already created at SA installation, packets are expected to match the existing ESP session.

The Group VPNv2 decryption path is as follows:

1. Packet received by the Packet Forwarding Engine undergoes a fragmentation check. If the packet is fragmented, it is assembled for further processing.

2. After packet assembling or if the packet is not fragmented, a zero source and destination IP is used in the 5-tuple decrypt flow lookup. If a match is found, the packet is further processed and transmitted.
3. If the decrypt flow lookup fails, the packet is checked against an SPI flow with zero source and destination IP.
4. If the SPI flow lookup fails, the packet is dropped.
5. If an SPI flow match is found, a decrypt flow is created to avoid the SPI flow lookup for subsequent packets.

Configuring a Routing Instance for Group VPNv2

Routing instances are supported for both control and data traffic. To enable routing instance support on control plane traffic for a group member to reach the GC/KS in a given VRF routing instance, add the `routing-instance` statement at the `[edit security group-vpn member ike gateway` *`gateway-name`* `local-address` *`address`*`]` hierarchy level.

No additional CLI is required to support a routing instance for data-plane packets, as it is determined based on the media interface on which the service set is applied.

Establishing Multiple Groups, Policies, and SAs

Junos OS provides support for one Group VPN per service set in Group VPNv2. However, multiple service sets can be created to support multiple groups in a routing instance. Multiple SAs can be configured per group. However, multiple policies for the same traffic key/SPI is not supported. If the server sends two policies for the same TEK, then they must be paired to be accepted, for instance, A-B and B-A, where A and B are IP addresses or subnets. If multiple unpaired policies for a given TEK are received, registration fails and a system log message is generated.

Connecting with Multiple Cooperative GC/KSs

For a group member to work with a GC/KS in the cooperative mode, the configuration is extended to allow a maximum of four servers in the server list.

During rekeying when using the `groupkey-pull` method, the group member tries to connect to the GC/KS. When the connection to the GC/KS fails, the group member tries to reconnect to the GC/KS. After three retries with an interval of 10 seconds, if the connection to the GC/KS is not restored, the group member tries to establish a connection with the next available server on the server list. This process is repeated until the group member connects to a GC/KS. During this time, the unexpired GDOI SAs on the group members are not cleaned up, so Group VPN traffic is not affected. The time gap between rekeying and hard lifetime expiry provides sufficient time for the group members to connect to the next available server, in such cases.

Implementing IP Delivery Delay Detection Protocol (Time-Based Anti-Replay Protection)

There is no configuration needed to implement the IP Delivery Delay Detection Protocol. MX Series group members get the replay window size to use as a part of the GAP payload in push or pull messages from the key server. If the received window size is 0, time-based anti-replay protection is disabled.

If IP Delivery Delay Detection Protocol is enabled, the sender adds its current timestamp and encrypts the packet. The receiver decrypts the packet and compares its current time with the timestamp in the packet. Packets that fall outside of the window size are dropped. Because of this, all group members should have their clocks synchronized using Network Time Protocol (NTP).

IP Delivery Delay Detection Protocol times are measured in seconds. See IP Delivery Delay Detection Protocol-draft-weis-delay-detection-01 for more information.

NOTE: All latency issues associated with NTP also apply within IP Delivery Delay Detection Protocol. Thus, a minimum window size of 1 second is recommended.

Changing Group VPNv2 Configuration

Most Group VPNv2 configuration changes result in deleting both existing SAs and re-registration. This triggers both phase 1 and SA download with new traffic keys.

Bypassing Group VPNv2 Configuration

If certain traffic like a routing protocol needs to bypass a Group VPN in Group VPNv2, a service filter needs to be configured on the interface on which the service set is applied. Packets matching the service filter do not come to the PIC for service processing and are directly forwarded to the Routing Engine.

Sample Service Set Filter Configuration

```
[edit interfaces]
interface-name {
    unit 0 {
        family inet {
            service {
                input {
                    service-set service-set-name service-filter filter-name;
                }
                output {
                    service-set service-set-name service-filter filter-name;
                }
            }
        }
    }
}
```

Implementing Partial Fail-open on MX Series Member Routers

By default, packets are dropped if a group member router is unable to get SAs from the GC/KS due to loss of connectivity. If you want to allow some traffic to pass unencrypted in the event of a communication failure between the group member and the GC/KS, you must configure a `fail-open` rule at the [`edit security group-vpn member ipsec vpn` *vpn-name*] hierarchy level.

Fail-open rules will be applied to the traffic only in the event of loss of server connectivity. Fail-open rules will be deactivated once connectivity is restored and keys are received from the GC/KS.

Sample Fail-Open Rule Configuration

```
[edit security group-vpn member ipsec vpn vpn-name]
fail-open {
    rule rule-name{
        source-address source-ip-address
            destination-address destination-ip-address}
        }
    }
```

A maximum of 10 fail-open rules can be configured for any given group.

Supported GDOI IPsec Parameters

Every GDOI group has a unique ID. It is used as a common base between GC/KS and the group member to communicate about group SAs and group keys.

During the registration process, the GC/KS sends Security Association Transport Encryption Keys (SA TEKs) to the group members. All parameters regarding the whole group security policy are configured on the GC/KS. The SA TEK is used by the group members to protect the traffic exchanged among each other. Table 5 on page 264 shows the parameters of the SA TEK.

Table 5: SA TEK Parameters

Parameters	Supported Values
Encryption	• DES-CBC • 3DES-CBC • AES-CBC 128 • AES-CBC 192 • AES-CBC 256
Integrity	• HMAC-MD5-96 • HMAC-SHA1-96 • HMAC-SHA-256-128
Lifetime	Any supported value

Besides the cryptographic algorithms, the traffic, which should be encrypted by the group members, is part of the SA TEK policy (traffic selector).

The following statements can be used on a Juniper Networks group member. Thus, the addresses have to be specified under the IKE hierarchy level. The enumeration is also prioritized. Thus, in the following example configuration, KS1 is contacted before KS2.

Sample GDOI IPsec Parameters Configuration

```
[edit security]
group-vpn {
    member {
        ike {
            gateway gateway-name {
                ike-policy policy-name;
                server-address <IP_KS1> <IP_KS2> <IP_KS3> <IP_KS4>;
                local-address <IP_GM> routing-instance routing-instance-name;
            }
        }
        ipsec {
            vpn vpn-group-name {
                ike-gateway gateway-name;
                fail-open {
                    rule rule-name {
                    source-address 198.51.100.1/24
                    destination-address 192.0.2.1/24
                    }
                }
                group group-ID;
                match-direction output;
            }
        }
    }
}
```

Supported GDOI IKEv1 Parameters

The group members use only IKEv1 during the registration process in the Group VPNv2 environment. Table 6 on page 266 provides an overview of the defined parameters of the IKEv1 SA.

Table 6: IKEv1 SA Parameters of Group Member

Parameter	Supported Values
Encryption	• DES-CBC • 3DES-CBC • AES-CBC 128 • AES-CBC 192 • AES-CBC 256
Authentication	Pre-shared key (minimum 20 signs)
Integrity	• MD5 • SHA1 • SHA256
Diffie-Hellman Group	• Group 1 • Group 2 • Group 5 • Group 14
Lifetime	Any supported value

The above-mentioned IKEv1 standards are configured as follows:

Applying Dynamic Policies

The `input` and `output` options under the `ipsec-group-vpn` statement specify if the dynamic policies received from the server are used when the interface on which the service set is applied is the incoming or outgoing interface. This provides flexibility to specify different rules in the incoming and outgoing directions.

Supporting TOS and DSCP

Type of service (TOS) and DiffServ Code Points (DSCP) bits are copied from the inner packet to the ESP packet.

Interoperability of Group Members

Cisco's implementation of GDOI is called Group Encryption Transport (GET) VPN. While Group VPNv2 in Junos OS and Cisco's GET VPN are both based on RFC 6407, *The Group Domain of Interpretation*, there are some implementation differences that you need to be aware of when deploying GDOI in a networking environment that includes both Juniper Networks security and routing devices and Cisco routers. For more information, see the current Junos OS release notes.

Group VPNv2 interoperability is as follows:

- Junos OS provides interoperability support with Cisco IOS GC/KS support.
- Junos OS does not provide support for Group VPNv2 interoperability with the SRX Series Group VPN server.

Table 7: Group VPNv2 Interoperability

Group Member	SRX Group Member	MX Group VPNv2 Member	Cisco Group Member	SRX GC	SRX KS	Cisco GC/KS
MX Group VPNv2 Member	No	Yes	Yes	No	Yes	Yes
SRX Group Member	Yes	No	No	Yes	Yes	Yes

Junos OS does not support the deny policy used on a Cisco GC/SK server to add an exception to the group policy. As a workaround, this can be done by configuring firewall rules on an MX Series group member. Also, Junos OS group members can work with the deny policy by not failing the negotiation and simply ignoring the contents. This allows system administrators to easily manage networks where both Cisco group members and Junos OS group members co-exist.

Group VPNv2 Limitations

Junos OS Group VPNv2 does not provide support for the following:

- Multicast push messages
- Multicast traffic
- GDOI SNMP MIBs
- Protocol and port in the policies sent by the server. The group member honors only the IP address/subnet specified in the policy.
- Multiple unpaired policies for the same traffic key/SPI
- Overlapping of both local and remote IP across routing instances in an IKE gateway configuration
- Overlapping Group VPNv2 policies that can result in mismatched SAs
- IPv6 for control and data traffic
- Co-existence of IPsec and Group VPN on the same service set
- Co-existence of services like NAT and ALG on the same service set. NAT and Group VPN can co-exist on different service sets. However, they cannot co-exist on the same service set.
- Site To Site (S2S) VPN and dynamic end point (DEP) VPN can co-exist with Group VPN on different service sets. However, they cannot co-exist on the same service set.
- Multiple groups on same service set
- Group member support with SRX Series GC/KS
- Group member support with SRX Series group member
- Logical Key Hierarchy (LKH)
- Graceful restart
- High availability
- Unified ISSU
- PKI support for authentication

RELATED DOCUMENTATION

Example: Configuring Group VPNs in Group VPNv2 on Routing Devices | **274**

Service Redundancy Daemon Overview

Configuring Group VPNs in Group VPNv2 on Routing Devices

You can configure an MX Series router with MS-MIC-16G and MS-MPC-PIC line cards to provide the Group VPNv2 member functionality support with one or more Cisco group controllers or key servers (GC/KSs). The group members can connect to a maximum of four Cisco GC/KSs with minimum interoperability with the cooperative servers.

The Group VPNv2 feature also provides system logging support for the Group VPNv2 functionality, and routing instance support for both control and data traffic.

Before you begin:

1. Configure the routers for network communication.
2. Configure the Cisco GC/KS.
3. Configure the group member device interfaces.
4. Configure a static route to reach the group server.

To configure a Group VPNv2 member, complete the following tasks:

1. In configuration mode, go to the following hierarchy level:

```
[edit]
user@GM1# edit security
```

2. Define the IKE proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal proposal-name
```

3. Configure the Phase 1 SA for the IKE proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal proposal-name  authentication-method pre-shared-
keys
user@GM1# set group-vpn member ike proposal proposal-name dh-group group
user@GM1# set group-vpn member ike proposal proposal-name authentication-algorithm sha1
user@GM1# set group-vpn member ike proposal proposal-name encryption-algorithm 3des-cbc
```

4. Define the IKE policy.

```
[edit security]
user@GM1# set group-vpn member ike policy policy-name mode main
user@GM1# set group-vpn member ike policy policy-name proposals proposal-name
user@GM1# set group-vpn member ike policy policy-name pre-shared-key ascii-text text
```

5. Set the remote gateways for the IKE gateway group.

```
[edit security]
user@GM1# set group-vpn member ike gateway gateway-group-name ike-policy policy-name
user@GM1# set group-vpn member ike gateway gateway-group-name server-address server-IP-address
user@GM1# set group-vpn member ike gateway gateway-group-name local-address server-facing-
interface-IP-address
```

NOTE: To configure a group member to connect to multiple group servers, add the IP address of all the servers to the remote IKE gateway group configuration.

For example,

```
[edit security]
user@GM1# set group-vpn member ike gateway gw-group1 server-address 203.0.113.0
user@GM1# set group-vpn member ike gateway gw-group1 server-address 203.0.113.1
```

6. Configure the group identifier and IKE gateway for the remote gateway group.

```
[edit security]
user@GM1# set group-vpn member ipsec vpn vpn-name ike-gateway gateway-group-name
user@GM1# set group-vpn member ipsec vpn vpn-name group group-ID
user@GM1# set group-vpn member ipsec vpn vpn-name match-direction output
```

7. In configuration mode, go to the following hierarchy level:

```
[edit]
user@GM1# edit services
```

8. Configure the service set for the remote gateway group.

```
[edit services]
user@GM1# set service-set service-set-name interface-service service-interface service-
interface
user@GM1# set service-set service-set-name ipsec-group-vpn vpn-name
```

NOTE: The service set has to be applied on the interface connecting to the other group member.

For example:

```
[edit interfaces]
user@GM1# set xe-0/3/1 unit 1 family inet service input service-set gvpn-service-set
user@GM1# set xe-0/3/1 unit 1 family inet service output service-set gvpn-service-set
```

9. Verify and commit the configuration.

   For example:

```
[edit security]
user@GM1# set group-vpn member ike proposal ike-proposal  authentication-method pre-shared-
keys
user@GM1# set group-vpn member ike proposal ike-proposal  dh-group group2
user@GM1# set group-vpn member ike proposal ike-proposal  authentication-algorithm sha1
user@GM1# set group-vpn member ike proposal ike-proposal  encryption-algorithm 3des-cbc
user@GM1# set group-vpn member ike policy ike-policy  mode main
user@GM1# set group-vpn member ike policy ike-policy proposals ike-proposal
user@GM1# set group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87u087-V4oz36"
user@GM1# set group-vpn member ike gateway gw-group1 ike-policy ike-policy
user@GM1# set group-vpn member ike gateway gw-group1 server-address 203.0.113.0
user@GM1# set group-vpn member ike gateway gw-group1 local-address 192.0.2.0
user@GM1# set group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
```

```
user@GM1# set group-vpn member ipsec vpn vpn-group1 group 1
user@GM1# set group-vpn member ipsec vpn vpn-group1 match-direction output
```

```
[edit services]
user@GM1# set service-set gvpn-service-set interface-service service-interface ms-4/0/0.1
user@GM1# set service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

```
[edit]
user@GM1# commit
commit complete
```

RELATED DOCUMENTATION

Example: Configuring Group VPNs in Group VPNv2 on Routing Devices | 274

Group VPN on AMS interfaces

Starting with Junos OS Release 18.4R1, Junos OS supports load balancing Group VPN services on Aggregated multiservices interface (AMS) interfaces. AMS is a bundle of multiple service interfaces combined together that functions as a single interface. Group VPN configured on AMS leverages the load balancing option in AMS to distribute traffic over the member interfaces that comprise the AMS interface. The Group VPN load balanced traffic follows the behavior defined by the hash key in the service set of the AMS interface.

Group VPN on AMS interface has the following limitation:

- Group VPN on AMS only supports load balancing.
- High Available is not supported on the AMS interface on Group VPNs.
- AMS is only supported on MX Series routers with more than one service PIC.

The following output shows a sample service set configuration on an AMS interface.

```
user@router1# show services service-set ss-gvpn400
interface-service {
    service-interface ams1.1;
    load-balancing-options {
        hash-keys {
            ingress-key [ source-ip protocol destination-ip ];
            egress-key [ destination-ip protocol source-ip ];
        }
    }
}
ipsec-group-vpn gvpn400;
```

For more information on configuring AMS, see Configuring Aggregated Multiservices Interfaces.

Use Case for Configuring Group VPNv2

NOTE: Group VPNv2 is the name of the Group VPN technology on MX5, MX10, MX40, MX80, MX104, MX240, MX480, and MX960 routers. Group VPNv2 is different from the Group VPN technology implemented on SRX Security Gateways. The term Group VPN is sometimes used in this document to refer to the technology in general, not to the SRX technology.

Today's networks support critical applications such as distributed computing, voice, and video over IP, which all require real-time branch-to-branch communication. With the increasing use of applications that are highly sensitive to latency and other delays, enterprise networks are tending toward meshed configurations where remote sites are directly connected to each other rather than through a central site. To provide the required infrastructure for supporting such applications and technologies, large service provider and enterprise networks implement any-to-any connectivity through IP VPNs and MPLS networks.

Although IP VPN and MPLS services separate enterprise traffic from the public Internet to provide security, there is an increasing need for enterprises to also encrypt private WANs that are built using service provider networks such as BGP over MPLS. In recent years, government regulations, such as the Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach-Bliley Act (GLBA), and Payment Card Industry Data Security Standard (PCI DSS), mandate encryption even over private IP networks.

Hub and spoke VPNs solve the problem of secure communication for enterprise WANs over the public Internet. For private IP and MPLS networks, Group VPNv2 addresses the above-mentioned issues across multiple sites using a group IPsec security paradigm.

Group VPNv2 on certain MX Series routers with MS-MIC-16G or MS-MPC-PIC line cards provides tunnel-less any-to-any encryption between devices in a private IP and MPLS network. Each device is a group member, using the same IPsec security association (SA) pair and keys provided by one or more Cisco Group Controllers or Key Servers (GC/KS).

Group VPNv2 is manageable and scalable. It provides any-to-any encrypted communication by replacing statically configured pair-wise IKE connections per peer with a dynamic group key management system. By providing encryption across private IP and MPLS networks, the Group VPNv2 implementation simplifies the management of secure branch-to-branch communication.

In addition to simplified key management, reduced latency, and improved any-to-any connectivity capabilities, Group VPNv2 also provides encryption for all WAN traffic, providing security compliance for internal governance and regulations.

Configuring Group VPNv2 provides an innovative and scalable solution to protect enterprise traffic as it passes through the meshed private WAN. Group VPNv2 optimizes network utilization and provides increased revenue by maintaining full-mesh connectivity and routing paths with existing MPLS backbones, eliminating the management and performance cost of a traditional full-mesh VPN network.

RELATED DOCUMENTATION

Group VPNv2 Overview | 245
Example: Configuring Group VPNs in Group VPNv2 on Routing Devices | 274

Example: Configuring Group VPNs in Group VPNv2 on Routing Devices

IN THIS SECTION

- Requirements | 275
- Overview | 275
- Configuration | 277
- Verification | 289
- Troubleshooting | 292

NOTE: Group VPNv2 is the name of the Group VPN technology on MX5, MX10, MX40, MX80, MX104, MX240, MX480, and MX960 routers. Group VPNv2 is different from the

Group VPN technology implemented on SRX Security Gateways. The term Group VPN is sometimes used in this document to refer to the technology in general, not to the SRX technology.

This example shows how to configure Group VPNs in Group VPNv2 to extend the IP Security (IPsec) architecture to support group security associations (GSAs) that are shared by a group of routers.

Requirements

This example uses the following hardware and software components:

- Two MX Series 5G Universal Routing Platforms with MS-MIC-16G or MS-MPC-PIC line cards
- Reachability to one or more Cisco Group Controllers or Key Servers (GC/KS)
- Junos OS Release 14.1 or later running on the MX Series routers

Before you begin:

1. Configure the routers for network communication.
2. Configure the Cisco GC/KS.
3. Configure the group member device interfaces.

Overview

IN THIS SECTION

- Topology | **276**

Starting with Junos OS Release 14.1, MX Series routers with MS-MIC-16G and MS-MPC-PIC line cards provide the Group VPNv2 member functionality support with one or more Cisco Group Controllers or Key Servers (GC/KS). The group members can connect to a maximum of four Cisco GC/KSs with minimum interoperability with the cooperative servers.

This feature also provides system logging support for Group VPNv2 functionality, and routing instance support for both control and data traffic.

Topology

In Figure 18 on page 276, a Group VPN is configured between a Cisco group server, GC/KS – and two group members, GM1 and GM2. The group members are connected to host devices.

In Figure 19 on page 276, a Group VPN is configured between GM1 and GM2, and GC/KS1 and GC/KS2 are the primary and secondary group servers, respectively.

Figure 18: Group VPN with Single GC/KS

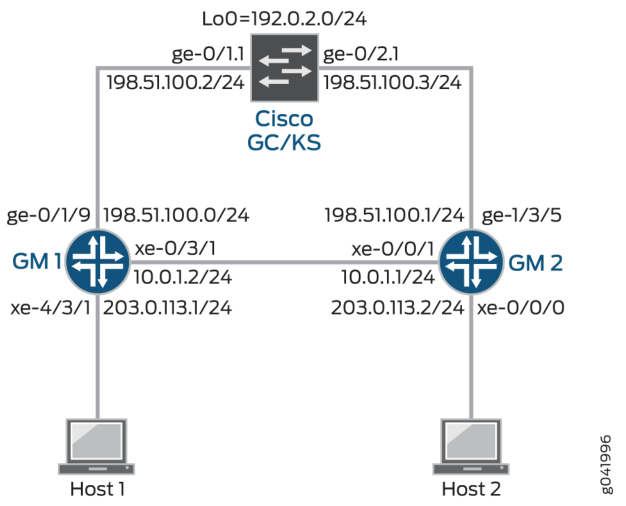
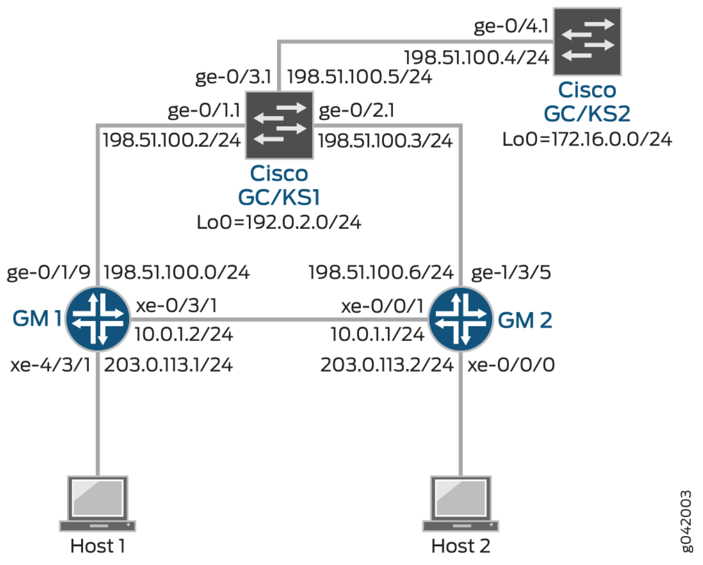


Figure 19: Group VPN with Multiple GC/KS



Configuration

IN THIS SECTION

- Configuring Group VPNv2 with a Single GC/KS | 277
- Configuring Group VPNv2 with Multiple GC/KS | 283

Configuring Group VPNv2 with a Single GC/KS

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 `commit` the configuration.

GM1

```
set interfaces ms-4/0/0 unit 1 family inet
set interfaces ge-0/1/9 vlan-tagging
set interfaces ge-0/1/9 unit 1 vlan-id 11
set interfaces ge-0/1/9 unit 1 family inet address 198.51.100.0/24
set interfaces xe-0/3/1 vlan-tagging
set interfaces xe-0/3/1 unit 1 vlan-id 1
set interfaces xe-0/3/1 unit 1 family inet service input service-set gvpn-service-set
set interfaces xe-0/3/1 unit 1 family inet service output service-set gvpn-service-set
set interfaces xe-0/3/1 unit 1 family inet address 10.0.1.2/24
set interfaces xe-4/3/1 unit 0 family inet address 203.0.113.1/24
set routing-options static route 192.0.2.0/24 next-hop 198.51.100.2
set routing-options static route 203.0.113.0/24 next-hop 10.0.1.1
set security group-vpn member ike proposal ike-proposal authentication-method pre-shared-keys
set security group-vpn member ike proposal ike-proposal dh-group group2
set security group-vpn member ike proposal ike-proposal authentication-algorithm sha1
set security group-vpn member ike proposal ike-proposal encryption-algorithm 3des-cbc
set security group-vpn member ike policy ike-policy mode main
set security group-vpn member ike policy ike-policy proposals ike-proposal
set security group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87uO87-V4oz36"
set security group-vpn member ike gateway gw-group1 ike-policy ike-policy
set security group-vpn member ike gateway gw-group1 server-address 192.0.2.0
set security group-vpn member ike gateway gw-group1 local-address 198.51.100.0
set security group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
set security group-vpn member ipsec vpn vpn-group1 group 1
set security group-vpn member ipsec vpn vpn-group1 match-direction output
set services service-set gvpn-service-set interface-service service-interface ms-4/0/0.1
set services service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

GM2

```
set interfaces ms-0/2/0 unit 1 family inet
set interfaces xe-0/0/0 unit 0 family inet address 203.0.113.2/24
set interfaces xe-0/1/1 vlan-tagging
set interfaces xe-0/1/1 unit 1 vlan-id 1
set interfaces xe-0/1/1 unit 1 family inet service input service-set gvpn-service-set
set interfaces xe-0/1/1 unit 1 family inet service output service-set gvpn-service-set
set interfaces xe-0/1/1 unit 1 family inet address 10.0.1.1/24
set interfaces ge-1/3/5 vlan-tagging
set interfaces ge-1/3/5 unit 1 vlan-id 11
set interfaces ge-1/3/5 unit 1 family inet address 198.51.100.1/24
set routing-options static route 192.0.2.0/24 next-hop 198.51.100.3
set routing-options static route 203.0.113.2/24 next-hop 10.0.1.2
set security group-vpn member ike proposal ike-proposal authentication-method pre-shared-keys
set security group-vpn member ike proposal ike-proposal dh-group group2
set security group-vpn member ike proposal ike-proposal authentication-algorithm sha1
set security group-vpn member ike proposal ike-proposal encryption-algorithm 3des-cbc
set security group-vpn member ike policy ike-policy  mode main
set security group-vpn member ike policy ike-policy proposals ike-proposal
set security group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87uO87-V4oz36"
set security group-vpn member ike gateway gw-group1 ike-policy ike-policy
set security group-vpn member ike gateway gw-group1 server-address 192.0.2.0
set security group-vpn member ike gateway gw-group1 local-address 198.51.100.0
set security group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
set security group-vpn member ipsec vpn vpn-group1 group 1
set security group-vpn member ipsec vpn vpn-group1 match-direction output
set services service-set gvpn-service-set interface-service service-interface ms-0/2/0.1
set services service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

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*.

To configure GM1:

1. Configure the Router GM1 interfaces.

```
[edit interfaces]
user@GM1# set ms-4/0/0 unit 1 family inet
user@GM1# set ge-0/1/9 vlan-tagging
user@GM1# set ge-0/1/9 unit 1 vlan-id 11
user@GM1# set ge-0/1/9 unit 1 family inet address 198.51.100.0/24
user@GM1# set xe-0/3/1 vlan-tagging
user@GM1# set xe-0/3/1 unit 1 vlan-id 1
user@GM1# set xe-0/3/1 unit 1 family inet service input service-set gvpn-service-set
user@GM1# set xe-0/3/1 unit 1 family inet service output service-set gvpn-service-set
user@GM1# set xe-0/3/1 unit 1 family inet address 10.0.1.2/24
user@GM1# set interfaces xe-4/3/1 unit 0 family inet address 203.0.113.1/24
```

2. Configure static routes to reach the group server and member 2.

```
[edit routing-options]
user@GM1# set static route 192.0.2.0/24 next-hop 198.51.100.2
user@GM1# set static route 203.0.113.0/24 next-hop 10.0.1.1
```

3. Define the IKE proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal ike-proposal
```

4. Configure the Phase 1 SA for ike-proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal ike-proposal  authentication-method pre-shared-
keys
user@GM1# set group-vpn member ike proposal ike-proposal dh-group group2
```

```
user@GM1# set group-vpn member ike proposal ike-proposal authentication-algorithm sha1
user@GM1# set group-vpn member ike proposal ike-proposal encryption-algorithm 3des-cbc
```

5. Define the IKE policy.

```
[edit security]
user@GM1# set group-vpn member ike policy ike-policy  mode main
user@GM1# set group-vpn member ike policy ike-policy proposals ike-proposal
user@GM1# set group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87u087-V4oz36"
```

6. Set the remote gateways for gw-group1.

```
[edit security]
user@GM1# set group-vpn member ike gateway gw-group1 ike-policy ike-policy
user@GM1# set group-vpn member ike gateway gw-group1 server-address 192.0.2.0
user@GM1# set group-vpn member ike gateway gw-group1 local-address 198.51.100.0
```

7. Configure the group identifier and IKE gateway for gw-group1.

```
[edit security]
user@GM1# set group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
user@GM1# set group-vpn member ipsec vpn vpn-group1 group 1
user@GM1# set group-vpn member ipsec vpn vpn-group1 match-direction output
```

8. Configure the service set for gw-group1.

```
[edit services]
user@GM1# set service-set gvpn-service-set interface-service service-interface ms-4/0/0.1
user@GM1# set service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

Results

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

GM1

```
user@GM1# show interfaces
ge-0/1/9 {
    vlan-tagging;
    unit 1 {
        vlan-id 11;
        family inet {
            address 198.51.100.0/24;
        }
    }
}
xe-0/3/1 {
    vlan-tagging;
    unit 1 {
        vlan-id 1;
        family inet {
            service {
                input {
                    service-set gvpn-service-set;
                }
                output {
                    service-set gvpn-service-set;
                }
            }
            address 10.0.1.2/24;
        }
    }
}
ms-4/0/0 {
    unit 1 {
        family inet;
    }
}
xe-4/3/1 {
    unit 0 {
        family inet {
            address 203.0.113.1/24;
        }
    }
}
```

```
user@GM1# show routing-options
static {
    route 192.0.2.0/24 next-hop 198.51.100.2;
    route 203.0.113.0/24 next-hop 10.0.1.1;
}
```

```
user@GM1# show security
group-vpn {
    member {
        ike {
            proposal ike-proposal {
                authentication-method pre-shared-keys;
                dh-group group2;
                authentication-algorithm sha1;
                encryption-algorithm 3des-cbc;
            }
            policy ike-policy {
                mode main;
                pre-shared-key ascii-text ""$9$QEni3/t1RSM87uO87-V4oz36"; ## SECRET-DATA
                proposals ike-proposal;
            }
            gateway gw-group1 {
                ike-policy ike-policy;
                server-address 192.0.2.0;
                local-address 198.51.100.0;
            }
        }
        ipsec {
            vpn vpn-group1 {
                ike-gateway gw-group1;
                group 1;
                match-direction output;
            }
        }
    }
}
```

```
user@GM1# show services
service-set gvpn-service-set {
    interface-service {
        service-interface ms-4/0/0.1;
    }
    ipsec-group-vpn vpn-group1;
}
```

Configuring Group VPNv2 with Multiple GC/KS

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 `commit` the configuration.

GM1

```
set interfaces ms-4/0/0 unit 1 family inet
set interfaces ge-0/1/9 vlan-tagging
set interfaces ge-0/1/9 unit 1 vlan-id 11
set interfaces ge-0/1/9 unit 1 family inet address 198.51.100.0/24
set interfaces xe-0/3/1 vlan-tagging
set interfaces xe-0/3/1 unit 1 vlan-id 1
set interfaces xe-0/3/1 unit 1 family inet service input service-set gvpn-service-set
set interfaces xe-0/3/1 unit 1 family inet service output service-set gvpn-service-set
set interfaces xe-0/3/1 unit 1 family inet address 10.0.1.2/24
set interfaces xe-4/3/1 unit 0 family inet address 203.0.113.1/24
set routing-options static route 192.0.2.0/24 next-hop 198.51.100.2
set routing-options static route 203.0.113.0/24 next-hop 10.0.1.1
set security group-vpn member ike proposal ike-proposal authentication-method pre-shared-keys
set security group-vpn member ike proposal ike-proposal dh-group group2
set security group-vpn member ike proposal ike-proposal authentication-algorithm sha1
set security group-vpn member ike proposal ike-proposal encryption-algorithm 3des-cbc
set security group-vpn member ike policy ike-policy  mode main
set security group-vpn member ike policy ike-policy proposals ike-proposal
set security group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
```

```
t1RSM87uO87-V4oz36"
set security group-vpn member ike gateway gw-group1 ike-policy ike-policy
set security group-vpn member ike gateway gw-group1 server-address 192.0.2.0
set security group-vpn member ike gateway gw-group1 server-address 172.16.0.0
set security group-vpn member ike gateway gw-group1 local-address 198.51.100.0
set security group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
set security group-vpn member ipsec vpn vpn-group1 group 1
set security group-vpn member ipsec vpn vpn-group1 match-direction output
set services service-set gvpn-service-set interface-service service-interface ms-4/0/0.1
set services service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

GM2

```
set interfaces ms-0/2/0 unit 1 family inet
set interfaces xe-0/0/0 unit 0 family inet address 203.0.113.2/24
set interfaces xe-0/1/1 vlan-tagging
set interfaces xe-0/1/1 unit 1 vlan-id 1
set interfaces xe-0/1/1 unit 1 family inet service input service-set gvpn-service-set
set interfaces xe-0/1/1 unit 1 family inet service output service-set gvpn-service-set
set interfaces xe-0/1/1 unit 1 family inet address 10.0.1.1/24
set interfaces ge-1/3/5 vlan-tagging
set interfaces ge-1/3/5 unit 1 vlan-id 11
set interfaces ge-1/3/5 unit 1 family inet address 198.51.100.1/24
set routing-options static route 192.0.2.0/24 next-hop 198.51.100.3
set routing-options static route 203.0.113.2/24 next-hop 10.0.1.2
set security group-vpn member ike proposal ike-proposal  authentication-method pre-shared-keys
set security group-vpn member ike proposal ike-proposal  dh-group group2
set security group-vpn member ike proposal ike-proposal  authentication-algorithm sha1
set security group-vpn member ike proposal ike-proposal  encryption-algorithm 3des-cbc
set security group-vpn member ike policy ike-policy  mode main
set security group-vpn member ike policy ike-policy proposals ike-proposal
set security group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87uO87-V4oz36"
set security group-vpn member ike gateway gw-group1 ike-policy ike-policy
set security group-vpn member ike gateway gw-group1 server-address 192.0.2.0
set security group-vpn member ike gateway gw-group1 server-address 172.16.0.0
set security group-vpn member ike gateway gw-group1 local-address 198.51.100.1
set security group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
set security group-vpn member ipsec vpn vpn-group1 group 1
set security group-vpn member ipsec vpn vpn-group1 match-direction output
set services service-set gvpn-service-set interface-service service-interface ms-0/2/0.1
set services service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

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*.

To configure GM1:

1. Configure the Router GM1 interfaces.

```
[edit interfaces]
user@GM1# set ms-4/0/0 unit 1 family inet
user@GM1# set ge-0/1/9 vlan-tagging
user@GM1# set ge-0/1/9 unit 1 vlan-id 11
user@GM1# set ge-0/1/9 unit 1 family inet address 198.51.100.0/24
user@GM1# set xe-0/3/1 vlan-tagging
user@GM1# set xe-0/3/1 unit 1 vlan-id 1
user@GM1# set xe-0/3/1 unit 1 family inet service input service-set gvpn-service-set
user@GM1# set xe-0/3/1 unit 1 family inet service output service-set gvpn-service-set
user@GM1# set xe-0/3/1 unit 1 family inet address 10.0.1.2/24
user@GM1# set xe-4/3/1 unit 0 family inet address 203.0.113.1/24
```

2. Configure static routes to reach the group server and member 2.

```
[edit routing-options]
user@GM1# set static route 192.0.2.0/24 next-hop 198.51.100.2
user@GM1# set static route 203.0.1.0/24 next-hop 10.0.1.1
```

3. Define the IKE proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal ike-proposal
```

4. Configure the Phase 1 SA for ike-proposal.

```
[edit security]
user@GM1# set group-vpn member ike proposal ike-proposal  authentication-method pre-shared-
```

```
keys
user@GM1# set group-vpn member ike proposal ike-proposal dh-group group2
user@GM1# set group-vpn member ike proposal ike-proposal authentication-algorithm sha1
user@GM1# set group-vpn member ike proposal ike-proposal encryption-algorithm 3des-cbc
```

5. Define the IKE policy.

```
[edit security]
user@GM1# set group-vpn member ike policy ike-policy  mode main
user@GM1# set group-vpn member ike policy ike-policy proposals ike-proposal
user@GM1# set group-vpn member ike policy ike-policy pre-shared-key ascii-text ""$9$QEni3/
t1RSM87uO87-V4oz36"
```

6. Set the remote gateways for gw-group1.

```
[edit security]
user@GM1# set group-vpn member ike gateway gw-group1 ike-policy ike-policy
user@GM1# set group-vpn member ike gateway gw-group1 server-address 192.0.2.0
user@GM1# set group-vpn member ike gateway gw-group1 server-address 172.16.0.0
user@GM1# set group-vpn member ike gateway gw-group1 local-address 198.51.100.0
```

7. Configure the group identifier and IKE gateway for gw-group1.

```
[edit security]
user@GM1# set group-vpn member ipsec vpn vpn-group1 ike-gateway gw-group1
user@GM1# set group-vpn member ipsec vpn vpn-group1 group 1
user@GM1# set group-vpn member ipsec vpn vpn-group1 match-direction output
```

8. Configure the service set for gw-group1.

```
[edit services]
user@GM1# set service-set gvpn-service-set interface-service service-interface ms-4/0/0.1
user@GM1# set service-set gvpn-service-set ipsec-group-vpn vpn-group1
```

Results

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

GM1

```
user@GM1# show interfaces
ge-0/1/9 {
    vlan-tagging;
    unit 1 {
        vlan-id 11;
        family inet {
            address 198.51.100.0/24;
        }
    }
}
xe-0/3/1 {
    vlan-tagging;
    unit 1 {
        vlan-id 1;
        family inet {
            service {
                input {
                    service-set gvpn-service-set;
                }
                output {
                    service-set gvpn-service-set;
                }
            }
            address 10.0.1.2/24;
        }
    }
}
ms-4/0/0 {
    unit 1 {
        family inet;
    }
}
xe-4/3/1 {
    unit 0 {
        family inet {
            address 203.0.113.1/24;
        }
    }
}
```

```
user@GM1# show routing-options
static {
    route 192.0.2.0/24 next-hop 198.51.100.2;
    route 203.0.113.0/24 next-hop 10.0.1.1;
}
```

```
user@GM1# show security
group-vpn {
    member {
        ike {
            proposal ike-proposal {
                authentication-method pre-shared-keys;
                dh-group group2;
                authentication-algorithm sha1;
                encryption-algorithm 3des-cbc;
            }
            policy ike-policy {
                mode main;
                pre-shared-key ascii-text ""$9$QEni3/t1RSM87uO87-V4oz36"; ## SECRET-DATA
                proposals ike-proposal;
            }
            gateway gw-group1 {
                ike-policy ike-policy;
                server-address [ 192.0.2.0 172.16.0.0 ];
                local-address 198.51.100.0;
            }
        }
        ipsec {
            vpn vpn-group1 {
                ike-gateway gw-group1;
                group 1;
                match-direction output;
            }
        }
    }
}
```

```
user@GM1# show services
service-set gvpn-service-set {
    interface-service {
        service-interface ms-4/0/0.1;
    }
    ipsec-group-vpn vpn-group1;
}
```

Verification

IN THIS SECTION

- Verifying the Group Member IKE SA | **289**
- Verifying the Group Member IPsec SA | **290**
- Verifying the Group Member IPsec Statistics | **291**

Confirm that the configuration is working properly.

Verifying the Group Member IKE SA

Purpose

Verify the IKE SAs on Router GM1.

Action

From operational mode, run the `show security group-vpn member ike security-associations detail` command.

```
user@GM1> show security group-vpn member ike security-associations detail
IKE peer 192.0.2.0, Index 2994970, Gateway Name: gw-group1
  Role: Initiator, State: UP
  Initiator cookie: 7fad16089a123bcd, Responder cookie: 536b33ffe89799de
  Exchange type: Main, Authentication method: Pre-shared-keys
  Local: 198.51.100.0:848, Remote: 192.0.2.0:848
  Lifetime: Expires in 175 seconds
  Peer ike-id: 192.0.2.0
  Xauth user-name: not available
  Xauth assigned IP: 0.0.0.0
  Algorithms:
   Authentication        : hmac-sha1-96
   Encryption            : 3des-cbc
   Pseudo random function: hmac-sha1
   Diffie-Hellman group  : DH-group-2
  Traffic statistics:
   Input  bytes  :                  752
   Output bytes  :                  716
   Input  packets:                    5
   Output packets:                    5
  Flags: IKE SA is created
  IPSec security associations: 0 created, 0 deleted
  Phase 2 negotiations in progress: 0
```

Meaning

Router GM1 has established the IKE SA with the GC/KS for the group.

Verifying the Group Member IPsec SA

Purpose

Verify the IPsec SAs on Router GM1.

Action

From operational mode, run the `show security group-vpn member ipsec security-associations detail` command.

```
user@GM1> show security group-vpn member ipsec security-associations detail
Virtual-system: root Group VPN Name: vpn-group1
  Local Gateway: 198.51.100.1, GDOI Server: 192.0.2.0
  Group Id: 1
  Rule Match Direction: output,  Tunnel-MTU: 1500
  Routing Instance: default
  DF-bit: clear
  Stats:
      Pull Succeeded            :   18
      Pull Failed               :   0
      Pull Timeout              :   0
      Pull Aborted              :   0
      Server Failover           :   0
      Delete Received           :   0
      Exceed Maximum Keys(4)    :   0
      Exceed Maximum Policies(1):   0
      Unsupported Algo          :   0
  Flags:
      Rekey Needed:   no

    List of policies received from server:
    Tunnel-id: 10001
      Source IP: ipv4_subnet(any:0,[0..7]=203.0.2.0/24)
      Destination IP: ipv4_subnet(any:0,[0..7]=203.0.1.0/24)

      Direction: bi-directional, SPI: e1c117c7
      Protocol: ESP, Authentication: sha1, Encryption: 3des
      Hard lifetime: Expires in 2526 seconds
      Lifesize Remaining:  Unlimited
      Soft lifetime: Expires in 2366 seconds
      Mode: Tunnel, Type: Group VPN, State: installed
      Anti-replay service: N/A
```

Meaning

Router GM1 has established the IPsec SA with the GC/KS.

Verifying the Group Member IPsec Statistics

Purpose

Verify the IPsec statistics on Router GM1.

Action

From operational mode, run the `show security group-vpn member ipsec statistics` command.

```
user@GM1> show security group-vpn member ipsec statistics
PIC: ms-0/2/0, Service set: gvpn-service-set
```

```
ESP Statistics:
  Encrypted bytes:              264
  Decrypted bytes:              264
  Encrypted packets:              3
  Decrypted packets:              3
AH Statistics:
  Input bytes:                    0
  Output bytes:                   0
  Input packets:                  0
  Output packets:                 0
Errors:
  AH authentication failures:     0
  ESP authentication failures:    0
  ESP decryption failures:        0
  Bad headers: 0, Bad trailers: 0
  Replay before window drops: 0, Replayed pkts: 0
  IP integrity errors: 0, Exceeds tunnel MTU: 0
  Rule lookup failures: 0, No SA errors: 0
  Flow errors: 0, Misc errors: 0
```

Meaning

`ESP Statistics` shows that packet flows have been encrypted and decrypted between the group members. Router GM1 has encrypted 3 packets and has received 3 decrypted packets from Router GM2.

Troubleshooting

IN THIS SECTION

- Negotiating the IKE SA | 293
- Establishing the IKE SA | 293
- Downloading the GDOI IPsec SA | 294
- Traffic Encryption and Decryption | 295
- Troubleshooting System Log Messages | 295

To troubleshoot the Group VPNv2 configuration, see:

Negotiating the IKE SA

Problem

The IKE SA negotiation is not triggered on the group member.

The output of the `show ike` and `show security group-vpn member ike security-associations` commands does not display the IKE negotiations.

Solution

To troubleshoot the IKE negotiation issue:

1. Check if the service interface status is up.

   Use `show interfaces terse | match ms` to check if the MS interface is down. An MS interface goes down when the PIC is rebooting.

2. Look for `Ignore gvpn vpn_name since it is inactive` in the log file `/var/log/gkmd`.

   Check if the Group VPN is referenced by any service set in the configuration.

   a. Enable `security group-vpn member ike traceoptions`.

   b. Look for the following system log messages in the trace log file:

      - `Dec 2 16:09:54 GVPN:iked_pm_gvpn_trigger called for gvpn200`
      - `Dec 2 16:09:54 GVPN:PM NULL for gvpn gvpn200`
      - `Dec 2 16:09:54 GVPN:Ignore gvpn gvpn200 since it is inactive`

This means either the service set is inactive or the service interface is down.

Establishing the IKE SA

Problem

The IKE SA is not getting established with the GC/KS.

In this scenario, the IKE SA state is down in the `show security group-vpn member ike security-associations` command output:

```
user@GM1> show security group-vpn member ike security-associations
Index   State  Initiator cookie  Responder cookie  Mode          Remote Address
       5295626 DOWN   2d47c125d2a9805e  0000000000000000  Main          192.0.2.2
```

Solution

To troubleshoot the IKE SA issue:

1. Check if the server address configured under `[edit security group-vpn member ike gateway]` is the correct one and is reachable.
2. Use the `ping` command between the remote devices to check network connectivity.
3. Check if the local address in the `group-vpn` configuration is also a configured address on any of the physical interfaces in the configuration.
4. Check if the IKE proposals match between the group member and the GC/KS.

   If there is a misconfiguration on the IKE SA negotiation, then do the following:

   a. Enable `security group-vpn member ike traceoption`.

   b. Look for the following message in the trace log file:
   `Dec 2 15:39:54 ikev2_fb_negotiation_done_isakmp: Entered IKE error code No proposal chosen (14), IKE SA 8dd7000 (neg 8dda800).`
5. Look for a `No proposal chosen` error in the log file `/var/log/gkmd`.

Downloading the GDOI IPsec SA

Problem

The GDOI IPsec SAs are not downloaded from the GC/KS.

In this scenario, the GDOI `groupkey-pull` with the configured GC/KS fails, and the `show security group-vpn member ipsec sa` command output does not display anything.

Solution

To troubleshoot the GDOI IPsec SA issue:

1. Check if the IKE SA has been established with the GC/KS.
2. Check if the group ID configured on the GC/KS and the group member match.
3. Look for any group SA installation failures or other failures in the log file `/var/log/gkmd`.

   Look for the following syslog messages to confirm use of an unsupported GDOI SA algorithm:

   - `Dec 2 15:32:49 simpleman gkmd[1701]: Failed to install SA because of unsupported algo(encr: 3des-cbc, auth : (null)) for SPI 0x6645cdb5 from server 192.0.2.1`
   - `Dec 2 15:32:49 simpleman gkmd[1701]: Member registration failed with key server 192.0.2.1 for group vpn gvpn200, reason SA unusable`

   Look for the following syslog messages to confirm use of unsupported GDOI policies:

   - `Dec 2 15:34:34 simpleman gkmd[1701]: Failed to install SA because of too many(2) policies for SPI 0x6951550c from server 192.0.2.1`
   - `Dec 2 15:34:34 simpleman gkmd[1701]: Member registration failed with key server 192.0.2.1 for group vpn gvpn200, reason SA unusable`

Traffic Encryption and Decryption

Problem

The CLI shows IPsec SAs as installed, but traffic does not go through the SAs.

In this scenario, traffic matching the rules received from the server fails to get encrypted or decrypted. The `show security group-vpn member ipsec statistics` command output displays a zero value for encrypt and decrypt packet count.

Solution

Look for `Rule lookup failures` counter in the error section of the CLI output.

Troubleshooting System Log Messages

Problem

System log messages are generated to record the different Group VPNv2 events.

Solution

To interpret the system log messages, refer to the following:

- `Dec 2 15:29:10 simpleman gkmd[1701]: Member registration succeeded with key server 192.0.2.1 for group vpn gvpn200`—GDOI pull was successful.
- `Dec 2 15:21:18 simpleman gkmd[1701]: Member registration failed with key server 192.0.2.1 for group vpn gvpn200, reason Timed out`—GDOI pull failed.
- `Dec 2 15:34:34 simpleman gkmd[1701]: Failed to install SA because of too many(2) policies for SPI 0x6951550c from server 192.0.2.1`—GDOI SA installation failed because of too many policies.
- `Dec 2 15:21:18 simpleman gkmd[1701]: Server 192.0.2.1 is unreachable for group vpn gvpn200`—Single GC/KS failed (Non-COOP).
- `Dec 2 15:51:49 simpleman gkmd[1701]: Current key server 192.0.2.1 is unreachable and will try registering with next Key Server 192.1.1.2 for group vpn gvpn200`—Particular GC/KS is not responding (COOP).
- `Dec 2 15:56:24 simpleman gkmd[1701]: All servers are unreachable for group vpn gvpn200`—None of the GC/KS are responding (COOP).
- `Dec 2 16:01:43 simpleman gkmd[1701]: Member re-registering with Key Server 192.0.2.1 for group-vpn gvpn200`—Member re-registration with the GC/KS.
- `Dec 2 16:01:43 simpleman gkmd[1701]: Creating TEK with SPI 0xb35200ac tunnel_id 10001 for group vpn gvpn200`—GDOI SA TEK creation was successful.
- `Dec 2 16:29:01 simpleman gkmd[1701]: Deleting TEK with SPI 0x6dba2a76 tunnel_id 10001 for group vpn gvpn200 and reason cleared from CLI`—GDOI SA TEK destroy was successful with reason.

  Different reasons for the GDOI SA TEK destroy are as follows:

  - Cleared from CLI
  - Hard lifetime expired
  - Too many TEKs
  - Configuration change
  - SA install error
  - Stale SA
  - Interface down

RELATED DOCUMENTATION

Use Case for Configuring Group VPNv2 | **273**

Group VPNv2 Overview | **245**

4

PART

Configuring Public Key Infrastructure

- Configuring Digital Certificate Validation | **298**
- Configuring a Device for Certificate Chains | **310**
- Managing Certificate Revocation | **326**

CHAPTER 17

Configuring Digital Certificate Validation

IN THIS CHAPTER

- Validate Digital Certificate on MX Series Devices | **298**
- Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device | **305**

Validate Digital Certificate on MX Series Devices

IN THIS SECTION

- Policy Validation | **298**
- Path Length Validation | **301**
- Key Usage | **302**
- Issuer and Subject Distinguished Name Validation | **303**

Staring in Junos OS Release 16.1R3, MX Series devices support digital certificate validation. During IKE negotiation, the PKI on an MX Series device validates X509 certificates received from VPN peers. The certificate validation performed is specified in RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Basic certificate and certificate chain validations include signature and date validation as well as revocation checks. This topic describes additional digital certificate validations performed by the PKI.

Policy Validation

X509 certificates can include optional policy validation fields. If a policy validation field is present, you must perform policy validation for the entire certificate chain including the EE certificate and intermediate CA certificates. Policy validation is not applicable to the root certificate. Policy validation ensures that the EE and intermediate CA certificates have a common policy. If no common policy exists for the certificate chain being validated, certificate validation fails.

Before performing policy validation, you must build a certificate chain containing the self-signed root certificate, intermediate CA certificate, and EE certificate. The policy validation starts with the intermediate CA certificate issued by the self-signed root certificate and continues through the EE certificate.

The following optional certificate fields are used for policy validation:

- **policy-oids**
- **requireExplicitPolicy**
- **skipCerts**

These fields are described in the following sections.

Policy OIDs Configured on MX Series Devices

In some situations, it might be desirable to only accept certificates with known policy object identifiers (OIDs) from peers. This optional configuration allows certificate validation to succeed only if the certificate chain received from the peer contains at least one policy OID that is configured on the MX Series device.

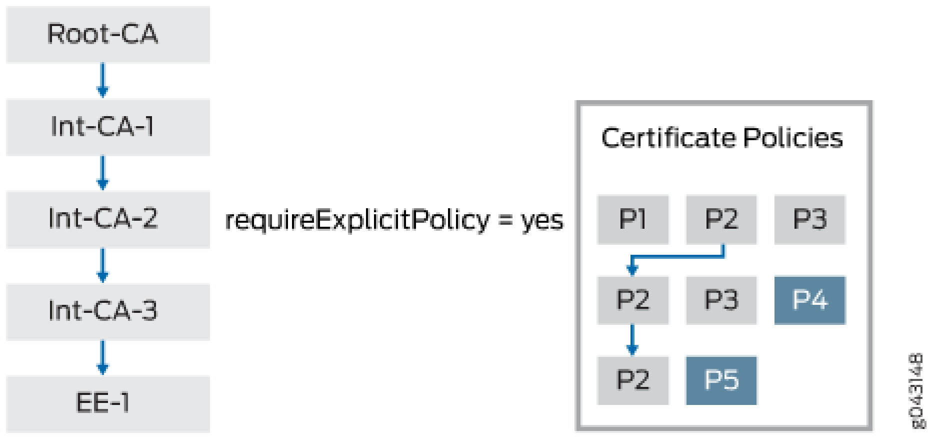
On the MX Series device, you must configure policy OIDs in an IKE policy with the `policy-oids` configuration statement at the [`edit security ike policy` *`policy-name`* `certificate`] hierarchy level. You can configure up to five policy OIDs. To validate a peer's certificate successfully, the peer's certificate chain must contain at least one of the policy OIDs configured on the MX Series device. Note that the **policy-oids** field in a certificate is optional. If you configure policy OIDs on the MX Series device but the peer's certificate chain does not contain any policy OIDs, certificate validation fails.

No Policy OIDs Configured on MX Series Devices

If no policy OID is configured on the MX Series device, policy validation starts whenever the **requireExplicitPolicy** field is encountered in the certificate chain. A certificate might contain one or more certificate policy OIDs. For successful policy validation, a common policy OID must be present in the certificate chain.

Figure 20 on page 300 shows a certificate chain that consists of certificates for a root CA, three intermediate CAs, and an EE. The CA certificate for Int-CA-2 contains the **requireExplicitPolicy** field; therefore, policy validation starts with Int-CA-2 and continues through EE-1. The certificate for Int-CA-2 contains policy OIDs P1, P2, and P3. The certificate for Int-CA-3 contains policy OIDs P2, P3, and P4. The certificate for EE-1 contains policy OIDs P2 and P5. Because the policy OID P2 is common to the certificates being validated, policy validation succeeds.

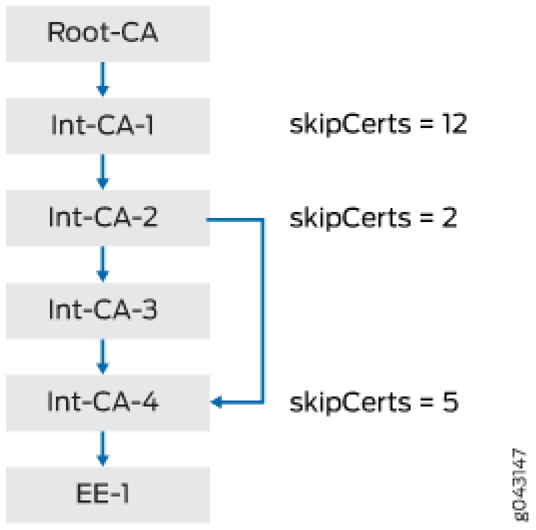
Figure 20: Policy Validation with requireExplicitPolicy Field



The optional **skipCerts** field in an intermediate CA certificate indicates the number of certificates, including the current CA certificate, that are to be excluded from policy validation. If **skipCerts** is 0, policy validation starts from the current certificate. If **skipCerts** is 1, the current certificate is excluded from policy validation. The value of the **skipCerts** field is checked in every intermediate CA certificate. If a **skipCerts** value is encountered that is lower than the current number of certificates being excluded, the lower **skipCerts** value is used.

Figure 21 on page 301 shows a certificate chain consisting of a root CA, four intermediate CAs, and an EE. The **skipCerts** value in Int-CA-1 is 12, which skips 12 certificates including the certificate for Int-CA-1. However, the **skipCerts** value is checked in every intermediate CA certificate in the chain. The **skipCerts** value in Int-CA-2 is 2, which is lower than 12, so now 2 certificates are skipped. The **skipCerts** value in Int-CA-4 is 5, which is greater than 2, so the Int-CA-4 **skipCerts** value is ignored.

Figure 21: Policy Validation with skipCerts Field



When policy OIDs are configured on the MX Series device, the certificate fields **requireExplicitPolicy** and **skipCerts** are ignored.

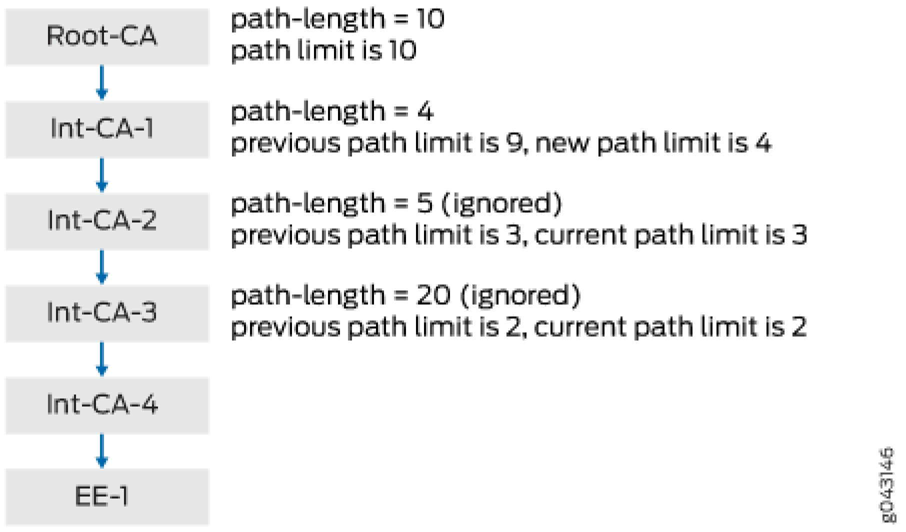
Path Length Validation

Certificate validation can involve a certificate chain that includes a root CA, one or more optional intermediate CAs, and an EE certificate. The number of intermediate CAs can grow depending upon the deployment scenario. Path length validation provides a mechanism to limit the number of intermediate certificates involved in certificate validation. **path-length** is an optional field in an X509 certificate. The value of **path-length** indicates the number of non-self-signed intermediate CA certificates allowed for certificate validation. The last certificate, which is generally the EE certificate, is not included in the path limit. If the root certificate contains a **path-length** value of 0, no intermediate CA certificates are allowed. If the **path-length** value is 1, there can be 0 or 1 intermediate CA certificates.

path-length can be present in multiple CA certificates in the certificate chain. The path length validation always begins with the self-signed root certificate. The path limit is decremented by 1 at each intermediate certificate in the chain. If an intermediate certificate contains a **path-length** value less than the current path limit, the new limit is enforced. On the other hand, if the **path-length** value is larger than the current path limit, it is ignored.

Figure 22 on page 302 shows a certificate chain that consists of a root CA, four intermediate CAs, and an EE. The **path-length** value in Root-CA is 10, therefore the initial path limit of non-self-signed intermediate CA certificates allowed for certificate validation is 10. At Int-CA-1, the path limit is 10-1 or 9. The **path-length** value in Int-CA-1 is 4, which is less than the path limit of 9, so the new path limit becomes 4. At Int-CA-2, the path limit is 4-1 or 3. The **path-length** value in Int-CA-2 is 5, which is larger than the path limit of 3, so it is ignored. At Int-CA-3, the path limit is 3-1 or 2. The **path-length** value in Int-CA-3 is 20, which is larger than the path limit of 2, so it is also ignored.

Figure 22: Path Length Validation


Root-CA
path-length = 10
path limit is 10
Int-CA-1
path-length = 4
previous path limit is 9, new path limit is 4
Int-CA-2
path-length = 5 (ignored)
previous path limit is 3, current path limit is 3
Int-CA-3
path-length = 20 (ignored)
previous path limit is 2, current path limit is 2
Int-CA-4
EE-1
g043146

Key Usage

The key usage field in an EE or CA certificate defines the purpose of the key contained in the certificate.

EE Certificates

For EE certificates, if the key usage field is present but the certificate does not contain **digitalSignature** or **nonrepudiation** flags, the certificate is rejected. If the key usage field is not present, then key usage is not checked.

CA Certificates

The key can be used for certificate or CRL signature validation. Because the PKI is responsible for both X509 certificate validation and CRL downloads, key usage must be checked before validating the certificate or CRL.

Certificate Signature Validation

The **keyCertSign** flag indicates that a CA certificate can be used for certificate signature validation. If this flag is not set, certificate validation is terminated.

CRL Signature Validation

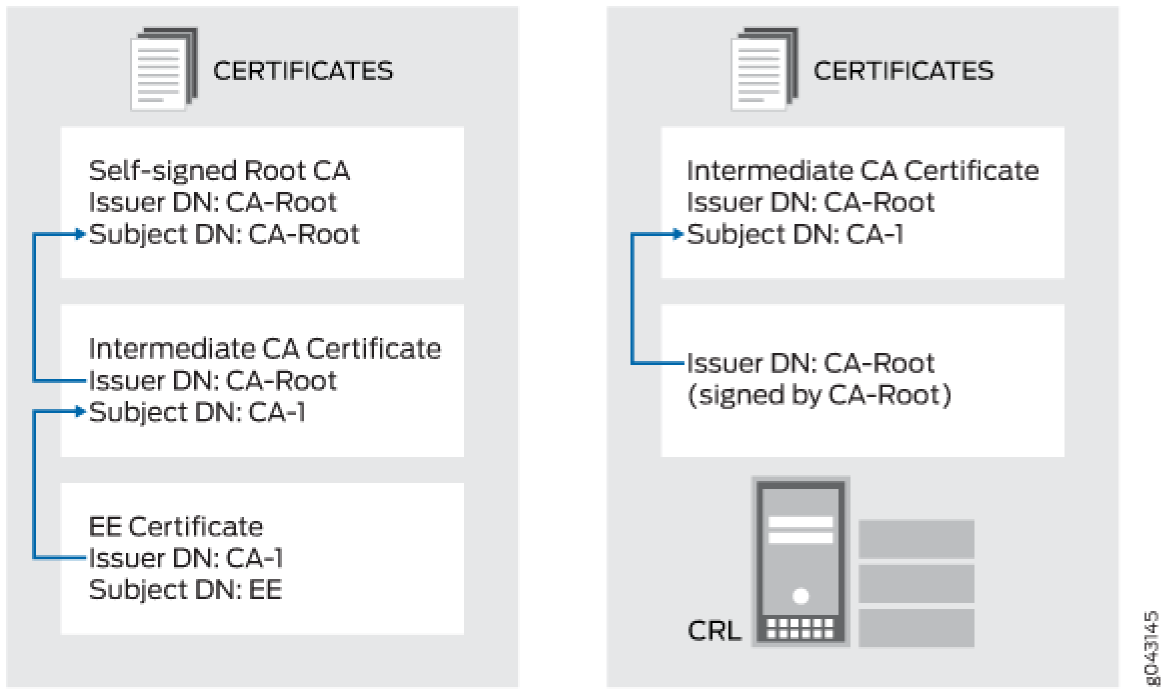
In Phase 1 negotiations, participants check the CRL to see if certificates received during an IKE exchange are still valid. The CRL is periodically downloaded for CA profiles configured with CRL as the certificate revocation check. Downloaded CRL files must be verified before they are downloaded into the device. One of the verification steps is to validate the CRL signature using a CA certificate. The downloaded CRL is signed with the CA certificate's private key and it must be verified with the CA certificate's public key stored in the device. The key usage field in the CA certificate must contain the **CRLSign** flag to verify the downloaded CRL. If this flag is not present, the CRL is discarded.

Issuer and Subject Distinguished Name Validation

Signature validation is performed for certificates received from a peer as well as for the CRL file downloaded from a CA server. Signature validation involves looking up the CA certificate in a CA database based on the issuer's DN in the certificate or the CRL being verified.

Figure 23 on page 304 shows the lookup for CA certificates based on the issuer DN. In the EE certificate, the issuer DN is CA-1, which is the subject DN of the intermediate CA certificate in the chain. In the intermediate CA certificate, the issuer DN is CA-Root, which is the subject DN of the self-signed Root-CA certificate in the chain. In the CRL, the issuer DN is CA-Root, which is the subject DN of the self-signed Root-CA certificate.

Figure 23: Issuer and Subject DN Validation



The lookup for the issuer or subject DN must follow these rules for attribute values:

- Attribute values encoded in different ASN.1 types (for example, PrintableString and BMPString) are assumed to represent different strings.
- Attribute values encoded in PrintableString types are not case-sensitive. These attribute values are compared after removing leading and trailing white spaces and converting internal substrings of one or more consecutive whites spaces to a single space.
- Attribute values encoded in types other than PrintableString are case-sensitive.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
16.1R3	Staring in Junos OS Release 16.1R3, MX Series devices support digital certificate validation.

Example: Improving Digital Certificate Validation by Configuring Policy OIDs on an MX Series Device

IN THIS SECTION

- Requirements | 305
- Overview | 305
- Configuration | 306
- Verification | 307

In some situations, it might be desirable to only accept certificates with known policy object identifiers (OIDs) from peers. This optional configuration allows certificate validation to succeed only if the certificate chain received from the peer contains at least one policy OID that is configured on the MX Series device. This example shows how to configure policy OIDs in the IKE policy on an MX Series device.

You must ensure that at least one of the policy OIDs configured on the MX Series device is included in a peer's certificate or certificate chain. Note that the **policy-oids** field in a peer's certificate is optional. If you configure policy OIDs in an IKE policy and, the peer's certificate chain does not contain any policy OIDs, certificate validation for the peer fails.

Requirements

Before you begin:

- Ensure that you are using Junos OS Release 16.1 or later for MX Series devices.
- Configure an IPsec VPN tunnel.

Overview

This example shows an IKE policy configuration where policy OIDs 2.16.840.1.101.3.1.48.2 and 5.16.40.1.101.3.1.55.2 are specified. The IKE policy ike_cert_pol references the IKE proposal ike_cert_prop, which is not shown. The local certificate on the MX Series device is lc-igloo-root.

Configuration

IN THIS SECTION

- Procedure | 306

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 configuration mode.

```
set services ipsec-vpn ike policy ike_cert_pol mode main
set services ipsec-vpn ike policy ike_cert_pol proposals ike_cert_prop
set services ipsec-vpn ike policy ike_cert_pol certificate local-certificate lc-igloo-root
set services ipsec-vpn ike policy ike_cert_pol certificate policy-oids 2.16.840.1.101.3.1.48.2
set services ipsec-vpn ike policy ike_cert_pol certificate policy-oids 5.16.40.1.101.3.1.55.2
```

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* in the CLI User Guide.

To configure policy OIDs for certificate validation:

- Configure the IKE policy:

  ```
  [edit services ipsec-vpn ike policy ike_cert_pol]
  user@host# set mode main
  user@host# set proposals ike_cert_prop
  user@host# set certificate local-certificate lc-igloo-root
  user@host# set certificate policy-oids 2.16.840.1.101.3.1.48.2
  user@host# set certificate policy-oids 5.16.40.1.101.3.1.55.2
  ```

Results

From configuration mode, confirm your configuration by entering the `show services ipsec-vpn ike policy ike_cert_pol` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show services ipsec-vpn ike policy ike_cert_pol
mode main;
proposals ike_cert_prop;
certificate {
    local-certificate lc-igloo-root;
    policy-oids [ 2.16.840.1.101.3.1.48.2 5.16.40.1.101.3.1.55.2 ];
}
```

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

Verification

IN THIS SECTION

- Verifying the CA Certificate | 307
- Verifying Policy OID Validation | 308

Confirm that the configuration is working properly.

Verifying the CA Certificate

Purpose

Display the CA certificate configured on the device.

Action

From the operational mode, enter the `show security pki ca-certificate ca-profile ca-tmp` command.

```
user@host> show security pki ca-certificate ca-profile ca-tmp detail
                   Certificate identifier: ca-tmp
                          Certificate version: 3
                          Serial number: 00000047
                          Issuer:
                            Organization: U.S. Government,
                            Organizational unit: DoD, Organizational unit: Testing, Country: US,
                            Common name: Trust Anchor
                          Subject:
                            Organization: U.S. Government,
                            Organizational unit: Dod, Organizational unit: Testing, Country: US,
                            Common name: CA1-PP.01.03
                          Subject string:
                            C=US, O=U.S. Government, OU=Dod, OU=Testing, CN=CA1-PP.01.03
                          Validity:
                            Not before: 07- 3-2015 10:54 UTC
                            Not after: 07- 1-2020 10:54 UTC

?Public key algorithm: rsaEncryption(1024 bits)
  30:81:89:02:81:81:00:cb:fd:78:0c:be:87:ac:cd:c0:33:66:a3:18
  9e:fd:40:b7:9b:bc:dc:66:ff:08:45:f7:7e:fe:8e:d6:32:f8:5b:75
  db:76:f0:4d:21:9a:6e:4f:04:21:4c:7e:08:a1:f9:3d:ac:8b:90:76
  44:7b:c4:e9:9b:93:80:2a:64:83:6e:6a:cd:d8:d4:23:dd:ce:cb:3b
  b5:ea:da:2b:40:8d:ad:a9:4d:97:58:cf:60:af:82:94:30:47:b7:7d
  88:c3:76:c0:97:b4:6a:59:7e:f7:86:5d:d8:1f:af:fb:72:f1:b8:5c
  2a:35:1e:a7:9e:14:51:d4:19:ae:c7:5c:65:ea:f5:02:03:01:00:01
Signature algorithm: sha1WithRSAEncryption
Certificate Policy:
  Policy Identifier = 2.16.840.1.101.3.1.48.2
Use for key: CRL signing, Certificate signing
Fingerprint:
  e0:b3:2f:2e:a1:c5:ee:ad:af:dd:96:85:f6:78:24:c5:89:ed:39:40 (sha1)
  f3:47:6e:55:bc:9d:80:39:5a:40:70:8b:10:0e:93:c5 (md5)
```

Verifying Policy OID Validation

Purpose

If the peer’s certificate is successfully validated, IKE and IPsec SA are established. If the validation of the peer’s certificate fails, no IKE SA is established.

Action

From the operational mode, enter the `show services ipsec-vpn ike security-associations` and `show services ipsec-vpn ipsec security-associations` commands.

```
user@host> show services ipsec-vpn ike security-associations
node0:
--------------------------------------------------------------------------
Index   State  Initiator cookie  Responder cookie  Mode           Remote Address
821765168 UP   88875c981252c1d8  b744ac9c21bde57e  IKEv2          192.0.2.1
1106977837 UP  1a09e32d1e6f20f1  e008278091060acb  IKEv2          198.51.100.0
```

```
user@host> show services ipsec-vpn ipsec security-associations
node0:
--------------------------------------------------------------------------
  Total active tunnels: 2
  ID    Algorithm       SPI      Life:sec/kb  Mon lsys Port  Gateway
  <213909506 ESP:aes-cbc-192/sha256 8cb9e40a 1295/ unlim - root 500 192.0.2.1
  >213909506 ESP:aes-cbc-192/sha256 8271d2b2 1295/ unlim - root 500 192.0.2.1
  <218365954 ESP:aes-cbc-192/sha256 d0153bc0 1726/ unlim - root 1495 198.51.100.0
  >218365954 ESP:aes-cbc-192/sha256 97611813 1726/ unlim - root 1495 198.51.100.0
```

Meaning

The `show services ipsec-vpn ike security-associations` command lists all active IKE Phase 1 SAs. If no SAs are listed, there was a problem with Phase 1 establishment. In this case, check for the PKID_CERT_POLICY_CHECK_FAIL message in the system logs. This message indicates that the peer's certificate chain does not contain a policy OID that is configured on the MX Series device. Check the **policy-oids** values in the peer's certificate chain with the values configured on the MX Series device.

It might also be that the peer's certificate chain does not contain any **policy-oids** fields, which are optional fields. If this is the case, certificate validation fails if there are any policy OIDs configured on the MX Series device.

CHAPTER 18

Configuring a Device for Certificate Chains

IN THIS CHAPTER

- IKE Authentication (Certificate-Based Authentication) | **310**
- Example: Configuring a Device for Peer Certificate Chain Validation | **313**

IKE Authentication (Certificate-Based Authentication)

IN THIS SECTION

- Multilevel Hierarchy for Certificate Authentication | **310**

Multilevel Hierarchy for Certificate Authentication

Certificate-based authentication is an authentication method supported on SRX Series Firewalls during IKE negotiation. In large networks, multiple certificate authorities (CAs) can issue end entity (EE) certificates to their respective end devices. It is common to have separate CAs for individual locations, departments, or organizations.

When a single-level hierarchy for certificate-based authentication is employed, all EE certificates in the network must be signed by the same CA. All firewall devices must have the same CA certificate enrolled for peer certificate validation. The certificate payload sent during IKE negotiation only contains EE certificates.

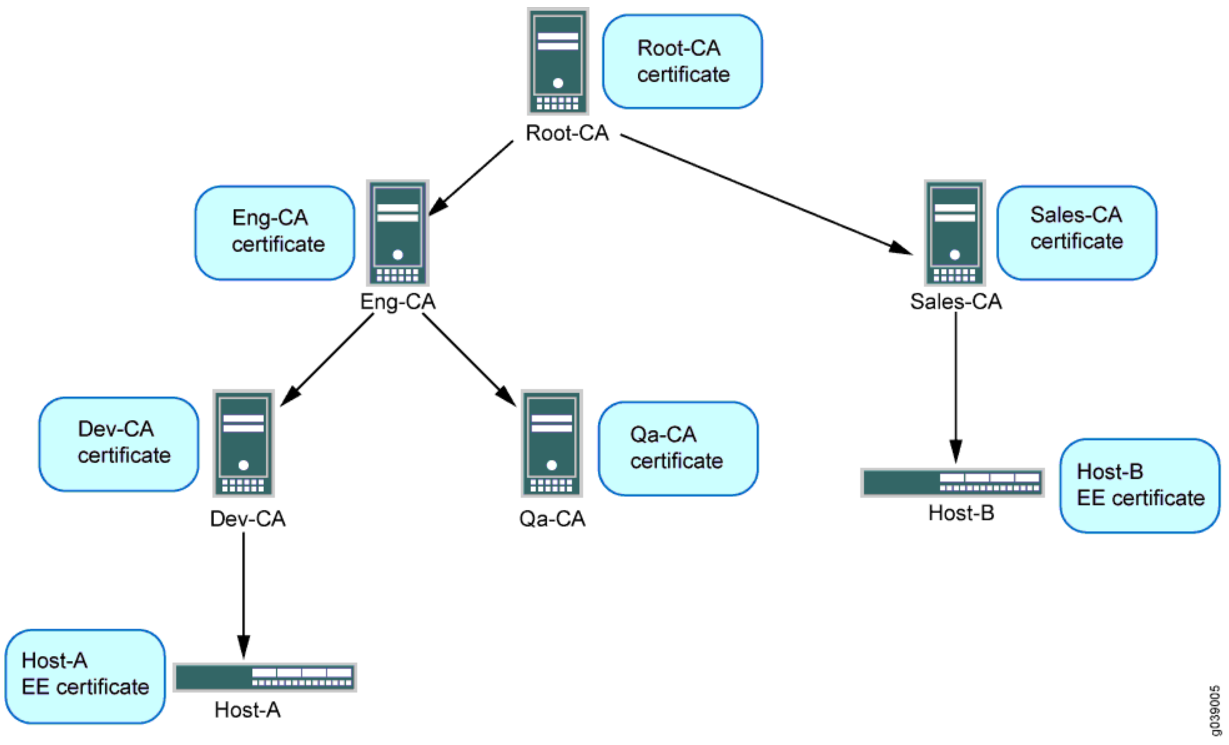
Alternatively, the certificate payload sent during IKE negotiation can contain a chain of EE and CA certificates. A *certificate chain* is the list of certificates required to validate a peer's EE certificate. The certificate chain includes the EE certificate and any CA certificates that are not present in the local peer.

The network administrator needs to ensure that all peers participating in an IKE negotiation have at least one common trusted CA in their respective certificate chains. The common trusted CA does not have to be the root CA. The number of certificates in the chain, including certificates for EEs and the topmost CA in the chain, cannot exceed 10.

Validation of a configured IKE peer can be done with a specified CA server or group of CA servers. With certificate chains, the root CA must match the trusted CA group or CA server configured in the IKE policy.

In the example CA hierarchy shown in Figure 24 on page 311, Root-CA is the common trusted CA for all devices in the network. Root-CA issues CA certificates to the engineering and sales CAs, which are identified as Eng-CA and Sales-CA, respectively. Eng-CA issues CA certificates to the development and quality assurance CAs, which are identified as Dev-CA and Qa-CA, respectively. Host-A receives its EE certificate from Dev-CA while Host-B receives its EE certificate from Sales-CA.

Figure 24: Multilevel Hierarchy for Certificate-Based Authentication



Each end device needs to be loaded with the CA certificates in its hierarchy. Host-A must have Root-CA, Eng-CA, and Dev-CA certificates; Sales-CA and Qa-CA certificates are not necessary. Host-B must have Root-CA and Sales-CA certificates. Certificates can be loaded manually in a device or enrolled using the Simple Certificate Enrollment Process (SCEP).

Each end device must be configured with a CA profile for each CA in the certificate chain. The following output shows the CA profiles configured on Host-A:

```
admin@host-A# show security
pki {
    ca-profile Root-CA {
        ca-identity Root-CA;
        enrollment {
            url “www.example.net/scep/Root/”;
        }
    }
    ca-profile Eng-CA {
        ca-identity Eng-CA;
        enrollment {
            url “www.example.net/scep/Eng/”;
        }
    }
    ca-profile Dev-CA {
        ca-identity Dev-CA;
        enrollment {
            url “www.example.net/scep/Dev/”;
        }
    }
}
```

The following output shows the CA profiles configured on Host-B:

```
admin@host-B# show security
pki {
    ca-profile Root-CA {
        ca-identity Root-CA;
        enrollment {
            url “www.example.net/scep/Root/”;
        }
    }
    ca-profile Sales-CA {
        ca-identity Sales-CA;
        enrollment {
            url “www.example.net/scep/Sales/”;
        }
    }
}
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.1R1	Starting with Junos OS Release 18.1R1, validation of a configured IKE peer can be done with a specified CA server or group of CA servers.

RELATED DOCUMENTATION

Basic Elements of PKI in Junos OS

Understanding Certificate Authority Profiles

Example: Configuring a Device for Peer Certificate Chain Validation

IN THIS SECTION

- Requirements | 313
- Overview | 314
- Configuration | 315
- Verification | 322
- IKE and IPsec SA Failure for a Revoked Certificate | 324

This example shows how to configure a device for certificate chains used to validate peer devices during IKE negotiation.

Requirements

Before you begin, obtain the address of the CA and the information they require (such as the challenge password) when you submit requests for local certificates.

Overview

IN THIS SECTION

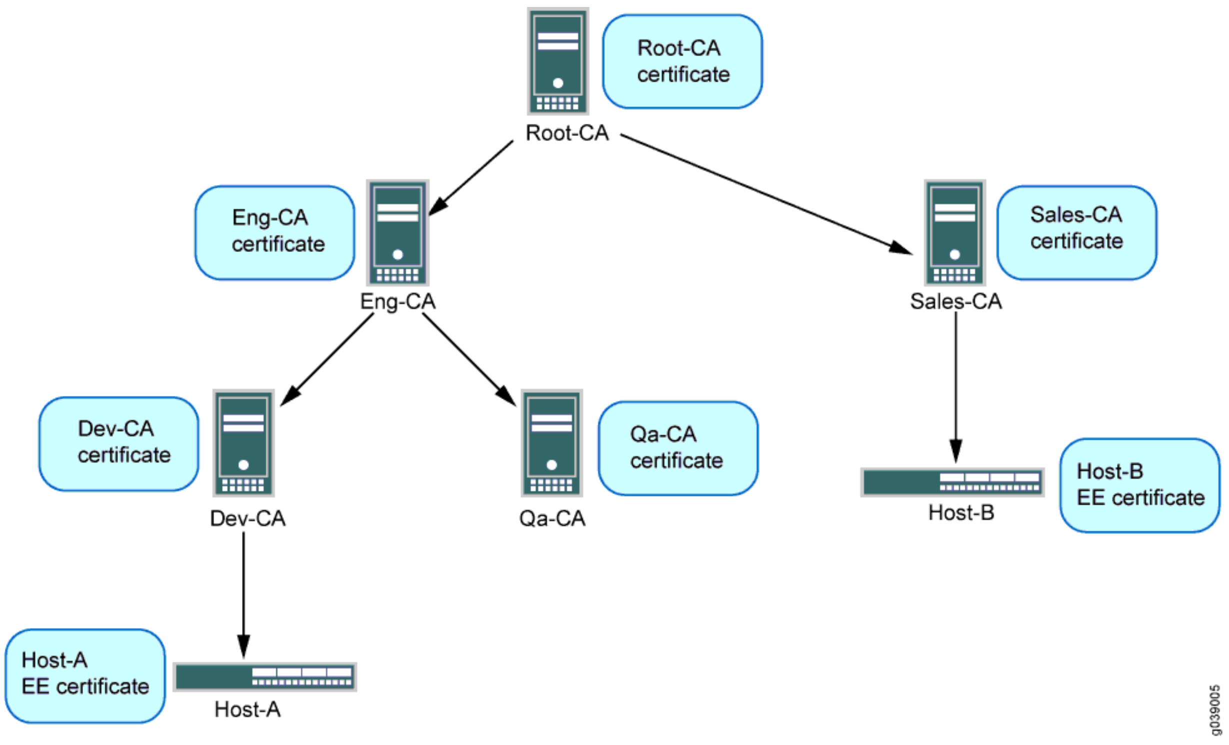
- Topology | **314**

This example shows how to configure a local device for certificate chains, enroll CA and local certificates, check the validity of enrolled certificates, and check the revocation status of the peer device.

Topology

This example shows the configuration and operational commands on Host-A, as shown in Figure 25 on page 314. A dynamic CA profile is automatically created on Host-A to allow Host-A to download the CRL from Sales-CA and check the revocation status of Host-B's certificate.

Figure 25: Certificate Chain Example



The IPsec VPN configuration for Phase 1 and Phase 2 negotiation is shown for Host-A in this example. The peer device (Host-B) must be properly configured so that Phase 1 and Phase 2 options are successfully negotiated and SA are established.

Configuration

IN THIS SECTION

- Configure CA Profiles | 315
- Enroll Certificates | 317
- Configure IPsec VPN Options | 320

To configure a device for certificate chains:

Configure CA Profiles

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 security pki ca-profile Root-CA ca-identity CA-Root
set security pki ca-profile Root-CA enrollment url http://10.157.88.230:8080/scep/Root/
set security pki ca-profile Root-CA revocation-check use-crl
set security pki ca-profile Eng-CA ca-identity Eng-CA
set security pki ca-profile Eng-CA enrollment url http://10.157.88.230:8080/scep/Eng/
set security pki ca-profile Eng-CA revocation-check use-crl
set security pki ca-profile Dev-CA ca-identity Dev-CA
set security pki ca-profile Dev-CA enrollment url http://10.157.88.230:8080/scep/Dev/
set security pki ca-profile Dev-CA revocation-check use-crl
```

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* in the CLI User Guide.

To configure CA profiles:

1. Create the CA profile for Root-CA.

```
[edit security pki]
user@host# set ca-profile Root-CA ca-identity CA-Root
user@host# set ca-profile Root-CA enrollment url http://10.157.88.230:8080/scep/Root/
user@host# set ca-profile Root-CA revocation-check use-crl
```

2. Create the CA profile for Eng-CA.

```
[edit security pki]
user@host# set ca-profile Eng-CA ca-identity Eng-CA
user@host# set ca-profile Eng-CA enrollment url http://10.157.88.230:8080/scep/Eng/
user@host# set ca-profile Eng-CA revocation-check use-crl
```

3. Create the CA profile for Dev-CA.

```
[edit security pki]
user@host# set ca-profile Dev-CA ca-identity Dev-CA
user@host# set ca-profile Dev-CA enrollment url http://10.157.88.230:8080/scep/Dev/
user@host# set ca-profile Dev-CA revocation-check use-crl
```

Results

From the configuration mode, confirm your configuration by entering the `show security pki` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security pki
ca-profile Root-CA {
    ca-identity Root-CA;
    enrollment {
        url "http:/;/10.157.88.230:8080/scep/Root/";
    }
    revocation-check {
        use-crl;
    }
}
ca-profile Eng-CA {
    ca-identity Eng-CA;
    enrollment {
        url "http:/;/10.157.88.230:8080/scep/Eng/";
    }
    revocation-check {
        use-crl;
    }
}
ca-profile Dev-CA {
    ca-identity Dev-CA;
    enrollment {
        url "http:/;/10.157.88.230:8080/scep/Dev/";
    }
    revocation-check {
        use-crl;
    }
}
```

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

Enroll Certificates

Step-by-Step Procedure

To enroll certificates:

1. Enroll the CA certificates.

   ```
   user@host> request security pki ca-certificate enroll ca-profile Root-CA
   ```

   ```
   user@host> request security pki ca-certificate enroll ca-profile Eng-CA
   ```

   ```
   user@host> request security pki ca-certificate enroll ca-profile Dev-CA
   ```

   Type **yes** at the prompts to load the CA certificate.

2. Verify that the CA certificates are enrolled in the device.

```
user@host> show security pki ca-certificate ca-profile Root-CA
Certificate identifier: Root-CA
        Issued to: Root-CA, Issued by: C = us, O = juniper, CN = Root-CA
        Validity:
          Not before: 07- 3-2015 10:54 UTC
          Not after: 07- 1-2020 10:54 UTC
        Public key algorithm: rsaEncryption(2048 bits)
```

```
user@host> show security pki ca-certificate ca-profile Eng-CA
Certificate identifier: Eng-CA
        Issued to: Eng-CA, Issued by: C = us, O = juniper, CN = Root-CA
        Validity:
          Not before: 07- 3-2015 10:54 UTC
          Not after: 07- 1-2020 10:54 UTC
        Public key algorithm: rsaEncryption(2048 bits)
```

```
user@host> show security pki ca-certificate ca-profile Dev-CA
        Certificate identifier: Dev-CA
        Issued to: Dev-CA, Issued by: C = us, O = juniper, CN = Eng-CA
        Validity:
          Not before: 07- 3-2015 10:54 UTC
          Not after: 07- 1-2020 10:54 UTC
        Public key algorithm: rsaEncryption(2048 bits)
```

3. Verify the validity of the enrolled CA certificates.

```
user@host> request security pki ca-certificate verify ca-profile Root-CA
CA certificate Root-CA verified successfully
```

```
user@host> request security pki ca-certificate verify ca-profile Eng-CA
CA certificate Eng-CA verified successfully
```

```
user@host> request security pki ca-certificate verify ca-profile Dev-CA
CA certificate Dev-CA verified successfully
```

4. Enroll the local certificate.

```
user@host> request security pki local-certificate enroll certificate-id        Host-A ca-
profile Dev-CA challenge-password juniper domain-name host-a.company.net email host-
a@company.net subject DC=juniper,CN=Host-A,       OU=DEV,O=PKI,L=Sunnyvale,ST=CA,C=US
```

5. Verify that the local certificate is enrolled in the device.

```
user@host> show security pki local-certificate
Issued to: Host-A, Issued by: C = us, O = juniper, CN = Dev-CA
        Validity:
          Not before: 07- 3-2015 10:54 UTC
          Not after: 07- 1-2020 10:54 UTC
        Public key algorithm: rsaEncryption(1024 bits)
```

6. Verify the validity of the enrolled local certificate.

```
user@host> request security pki local-certificate verify certificate-id Host-A
Local certificate Host-A verification success
```

7. Check the CRL download for configured CA profiles.

```
user@host> show security pki crl
     CA profile: Root-CA
```

```
      CRL version: V00000001
      CRL issuer: C = us, O = juniper, CN = Root-CA
      Effective date: 09- 9-2015 13:08
      Next update: 09-21-2015 02:55

    CA profile: Eng-CA
      CRL version: V00000001
      CRL issuer: C = us, O = juniper, CN = Eng-CA
      Effective date: 08-22-2015 17:46
      Next update: 10-24-2015 03:33

    CA profile: Dev-CA
      CRL version: V00000001
      CRL issuer: C = us, O = juniper, CN = Dev-CA
      Effective date: 09-14-2015 21:15
      Next update: 09-26-2012 11:02
```

Configure IPsec VPN Options

CLI Quick Configuration

To quickly configure IPsec VPN options, 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 services ipsec-vpn ike proposal ike_cert_prop_01 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_cert_prop_01 dh-group group5
set services ipsec-vpn ike proposal ike_cert_prop_01 authentication-algorithm sha1
set services ipsec-vpn ike proposal ike_cert_prop_01 encryption-algorithm aes-256-cbc
set services ipsec-vpn ike policy ike_cert_pol_01 mode main
set services ipsec-vpn ike policy ike_cert_pol_01 proposals ike_cert_prop_01
set services ipsec-vpn ike policy ike_cert_pol_01 certificate local-certificate Host-A
set services ipsec-vpn ipsec proposal ipsec_prop_01 protocol esp
set services ipsec-vpn ipsec proposal ipsec_prop_01 authentication-algorithm hmac-sha1-96
set services ipsec-vpn ipsec proposal ipsec_prop_01 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_prop_01 lifetime-seconds 300
set services ipsec-vpn ipsec policy ipsec_pol_01 proposals ipsec_prop_01
set services ipsec-vpn ipsec vpn ipsec_cert_vpn_01 ike ipsec-policy ipsec_pol_01
```

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* in the CLI User Guide.

To configure IPsec VPN options:

1. Configure Phase 1 options.

```
[edit services ipsec-vpn ike proposal ike_cert_prop_01]
user@host# set authentication-method rsa-signatures
user@host# set dh-group group5
user@host# set authentication-algorithm sha1
user@host# set encryption-algorithm aes-256-cbc
[edit services ipsec-vpn ike policy ike_cert_pol_01]
user@host# set mode main
user@host# set proposals ike_cert_prop_01
user@host# set certificate local-certificate Host-A
```

2. Configure Phase 2 options.

```
[edit services ipsec-vpn ipsec proposal ipsec_prop_01]
user@host# set protocol esp
user@host# set authentication-algorithm hmac-sha1-96
user@host# set encryption-algorithm 3des-cbc
user@host# set lifetime-seconds 300
[edit services ipsec-vpn ipsec policy ipsec_pol_01]
user@host# set proposals ipsec_prop_01
[edit services ipsec-vpn ipsec vpn ipsec_cert_vpn_01]
user@host# set ike ipsec-policy ipsec_pol_01
```

Results

From the configuration mode, confirm your configuration by entering the `show security ike` and `show security ipsec` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show services ipsec-vpn ike
proposal ike_cert_prop_01 {
    authentication-method rsa-signatures;
    dh-group group5;
    authentication-algorithm sha1;
    encryption-algorithm aes-256-cbc;
}
    policy ike_cert_pol_01 {
        mode main;
        proposals ike_cert_prop_01;
        certificate {
            local-certificate Host-A;
        }
    }
[edit]
user@host# show services ipsec-vpn ipsec
proposal ipsec_prop_01 {
    protocol esp;
    authentication-algorithm hmac-sha1-96;
    encryption-algorithm 3des-cbc;
    lifetime-seconds 300;
}
    policy ipsec_pol_01 {
        proposals ipsec_prop_01;
    }
```

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

Verification

IN THIS SECTION

- Verifying IKE Phase 1 Status | **323**
- Verify IPsec Phase 2 Status | **323**

If certificate validation is successful during IKE negotiation between peer devices, both IKE and IPsec SA are established.

Verifying IKE Phase 1 Status

Purpose

Verify the IKE Phase 1 status.

Action

Enter the **show services ipsec-vpn ike security-associations** command from the operational mode.

```
user@host> show services ipsec-vpn ike security-associations

Remote Address  State         Initiator cookie  Responder cookie  Exchange type
192.0.2.0        Matured       63b3445edda507fb  2715ee5895ed244d  Main
```

Verify IPsec Phase 2 Status

Purpose

Verify the IPsec Phase 2 status.

Action

Enter the **show services ipsec-vpn ipsec security-associations** command from the operational mode.

```
user@host> show services ipsec-vpn ipsec security-associations

Service set: ips_ss1, IKE Routing-instance: default

  Rule: vpn_rule_ms_2_2_01, Term: term11, Tunnel index: 1
  Local gateway: 10.0.1.2, Remote gateway: 172.16.0.0
  IPSec inside interface: ms-2/2/0.1, Tunnel MTU: 1500
  UDP encapsulate: Disabled, UDP Destination port: 0
    Direction SPI         AUX-SPI     Mode       Type     Protocol
    inbound   2151932129  0           tunnel     dynamic  ESP
    outbound  4169263669  0           tunnel     dynamic  ESP
```

IKE and IPsec SA Failure for a Revoked Certificate

IN THIS SECTION

● Checking for Revoked Certificates | 324

Checking for Revoked Certificates

Problem

If certificate validation fails during IKE negotiation between peer devices, check to make sure that the peer's certificate has not been revoked. A dynamic CA profile enables the local device to download the CRL from the peer's CA and check the revocation status of the peer's certificate. To enable dynamic CA profiles, the `revocation-check crl` option must be configured on a parent CA profile.

Solution

To check the revocation status of a peer's certificate:

1. Identify the dynamic CA profile that will show the CRL for the peer device by entering the **show security pki crl** command from operational mode.

```
user@host> show security pki crl
     CA profile: Root-CA
        CRL version: V00000001
        CRL issuer: C = us, O = juniper, CN = Root-CA
        Effective date: 09- 9-2012 13:08
        Next update: 09-21-2012 02:55

      CA profile: Eng-CA
        CRL version: V00000001
        CRL issuer: C = us, O = juniper, CN = Eng-CA
        Effective date: 08-22-2012 17:46
        Next update: 10-24-2015 03:33

      CA profile: Dev-CA
        CRL version: V00000001
        CRL issuer: C = us, O = juniper, CN = Dev-CA
        Effective date: 09-14-2012 21:15
```

```
          Next update: 09-26-2012 11:02

        CA profile: dynamic-001
          CRL version: V00000001
          CRL issuer: C = us, O = juniper, CN = Sales-CA
          Effective date: 09-14-2012 21:15
          Next update: 09-26-2012 11:02
```

The CA profile `dynamic-001` is automatically created on Host-A so that Host-A can download the CRL from Host-B's CA (Sales-CA) and check the revocation status of the peer's certificate.

2. Display CRL information for the dynamic CA profile by entering the **show security pki crl ca-profile dynamic-001 detail** command from the operational mode.

   Enter

```
user@host> show security pki crl ca-profile dynamic-001 detail
    CA profile: dynamic-001
      CRL version: V00000001
        CRL issuer: C = us, O = juniper, CN = Sub11
        Effective date: 09-19-2012 17:29
        Next update: 09-20-2012 01:49
        Revocation List:
          Serial number             Revocation date
          10647C84                  09-19-2012 17:29 UTC
```

   Host-B's certificate (serial number 10647084) has been revoked.

CHAPTER 19

Managing Certificate Revocation

IN THIS CHAPTER

- Online Certificate Status Protocol and Certificate Revocation Lists | 326
- Example: Improving Security by Configuring OCSP for Certificate Revocation Status | 329

Online Certificate Status Protocol and Certificate Revocation Lists

IN THIS SECTION

- Comparison of Online Certificate Status Protocol and Certificate Revocation List | 328

OCSP is used to check the revocation status of X509 certificates. OCSP provides revocation status on certificates in real time and is useful in time-sensitive situations such as bank transactions and stock trades.

The revocation status of a certificate is checked by sending a request to an OCSP server that resides outside of an SRX Series Firewall. Based on the response from the server, the VPN connection is allowed or denied. OCSP responses are not cached on SRX Series Firewalls.

The OCSP server can be the *certificate authority (CA)* that issues a certificate or a designated authorized responder. The location of the OCSP server can be configured manually or extracted from the certificate that is being verified. Requests are sent first to OCSP server locations that are manually configured in CA profiles with the `ocsp url` statement at the `[edit security pki ca-profile *profile-name* revocation-check]` hierarchy level; up to two locations can be configured for each CA profile. If the first configured OCSP server is not reachable, the request is sent to the second OCSP server. If the second OCSP server is not reachable, the request is then sent to the location in the certificate's AuthorityInfoAccess extension field. The `use-ocsp` option must also be configured, as *certificate revocation list (CRL)* is the default checking method.

SRX Series Firewalls accept only signed OCSP responses from the CA or authorized responder. The response received is validated using trusted certificates. The response is validated as follows:

1. The CA certificate enrolled for the configured CA profile is used to validate the response.
2. The OCSP response might contain a certificate to validate the OCSP response. The received certificate must be signed by a CA certificate enrolled in the SRX Series Firewall. After the received certificate is validated by the CA certificate, it is used to validate the OCSP response.

The response from the OCSP server can be signed by different CAs. The following scenarios are supported:

- The CA server that issues the EE certificate for a device also signs the OCSP revocation status response. The SRX Series Firewall verifies the OCSP response signature using the CA certificate enrolled in the SRX Series Firewall. After the OCSP response is validated, the certificate revocation status is checked.
- An authorized responder signs the OCSP revocation status response. The certificate for the authorized responder and the EE certificate being verified must be issued by the same CA. The authorized responder is first verified using the CA certificate enrolled in the SRX Series Firewall. The OCSP response is validated using the responder's CA certificate. The SRX Series Firewall then uses the OCSP response to check the revocation status of the EE certificate.
- There are different CA signers for the EE certificate being verified and the OCSP response. The OCSP response is signed by a CA in the certificate chain for the EE certificate being verified. (All peers participating in an IKE negotiation need to have at least one common trusted CA in their respective certificate chains.) The OCSP responder's CA is verified using a CA in the certificate chain. After validating the responder CA certificate, the OCSP response is validated using the responder's CA certificate.

To prevent replay attacks, a *nonce* payload can be sent in an OCSP request. Nonce payloads are sent by default unless it is explicitly disabled. If enabled, the SRX Series Firewall expects the OCSP response to contain a nonce payload, otherwise the revocation check fails. If OCSP responders are not capable of responding with a nonce payload, then the nonce payload must be disabled on the SRX Series Firewall.

In the normal course of business, certificates are revoked for various reasons. You might wish to revoke a certificate if you suspect that it has been compromised, for example, or when a certificate holder leaves the company.

You can manage certificate revocations and validations in two ways:

- Locally— This is a limited solution.
- By referencing a Certificate Authority (CA) certificate revocation list (CRL)— You can automatically access the CRL online at intervals you specify or at the default interval set by the CA.

In Phase 1 negotiations, SRX Series Firewall verifies the EE certificate received from the peer during an IKE exchange and uses the CRL to make sure the EE certificate is not revoked by its CA.

If a CRL is not loaded on the device and the peer certificate issuer is a trusted CA:

1. Junos OS retrieves the CRL through the configured LDAP or HTTP CRL locations (that is, the CRL Distribution Points (CDP)), if they are defined in the CA profile.

2. If the CRL Distribution Points is not configured in the CA profile, the device uses the CDP extension in a certificate issued by the CA (if present). The certificate issued by the CA can be a certificate enrolled by the administrator or received during the Phase 1 negotiation.

If the EE certificate is not issued by a root CA, the certificates of each intermediate CAs goes through the same verification and revocation check. The CRL of the root CA is used to check if the certificate issued by the root CA is revoked. If the CDP is not configured in the root CA profile, the device uses the CDP extension in the certificate issued by the CA (if present).

The CRL distribution point extension (.cdp) in an X509 certificate can be added to either an HTTP URL or an LDAP URL.

If the certificate does not contain a certificate distribution point extension, and you cannot automatically retrieve the CRL through Lightweight Directory Access Protocol (LDAP) or Hypertext Transfer Protocol (HTTP), you can retrieve a CRL manually and load that in the device.

Local certificates are being validated against certificate revocation list (CRL) even when CRL check is disabled. This can be stopped by disabling the CRL check through the Public Key Infrastructure (PKI) configuration. When CRL check is disabled, PKI will not validate local certificate against CRL.

Comparison of Online Certificate Status Protocol and Certificate Revocation List

Online Certificate Status Protocol (OCSP) and certificate revocation list (CRL) can both be used to check the revocation status of a certificate. There are advantages and disadvantages to each method.

- OCSP provides certificate status in real time, while CRL uses cached data. For time-sensitive applications, OCSP is the preferred approach.

- CRL checking is faster because lookup for certificate status is done on information cached on the VPN device. OCSP requires time to obtain the revocation status from an external server.

- CRL requires additional memory to store the revocation list received from a CRL server. OCSP does not require additional memory to save the revocation status of certificates.

- OCSP requires that the OCSP server be available at all times. CRL can use cached data to check the revocation status of certificates when the server is unreachable.

On MX Series and SRX Series Firewalls, CRL is the default method used to check the revocation status of a certificate.

RELATED DOCUMENTATION

Certificate Validation

Example: Improving Security by Configuring OCSP for Certificate Revocation Status

IN THIS SECTION

- Requirements | 329
- Overview | 329
- Configuration | 332
- Verification | 343

This example shows how to improve security by configuring two peers using the Online Certificate Status Protocol (OCSP) to check the revocation status of the certificates used in Phase 1 negotiations for the IPsec VPN tunnel.

Requirements

On each device:

- Obtain and enroll a local certificate. This can be done either manually or by using the Simple Certificate Enrollment Protocol (SCEP).
- Optionally, enable automatic renewal of the local certificate.
- Configure security policies to permit traffic to and from the peer device.

Overview

IN THIS SECTION

- Topology | 332

On both peers, a certificate authority (CA) profile Root is configured with the following options:

- CA name is Root.
- Enrollment URL is http://10.1.1.1:8080/scep/Root/. This is the URL where SCEP requests to the CA are sent.
- The URL for the OCSP server is http://10.157.88.56:8210/Root/.
- OCSP is used first to check the certificate revocation status. If there is no response from the OCSP server, then the certificate revocation list (CRL) is used to check the status. The CRL URL is http://10.1.1.1:8080/crl-as-der/currentcrl-45.crlid=45.
- The CA certificate received in an OCSP response is not checked for certificate revocation. Certificates received in an OCSP response generally have shorter lifetimes and a revocation check is not required.

Table 8 on page 330 shows the Phase 1 options used in this example.

Table 8: Phase 1 Options for OCSP Configuration Example

Option	Peer A	Peer B
IKE proposal	ike_policy_ms_2_2_0	ike_proposal_ms_2_0_0
Authentication method	rsa-signatures	rsa-signatures
DH group	group2	group2
Authentication algorithm	SHA 1	SHA 1
Encryption algorithm	3des-cbc	3des-cbc
Lifetime seconds	3000	3000
IKE policy	ike_policy_ms_2_2_0	ike_policy_ms_2_0_0
Mode	main	main
Proposal	ike_proposal_ms_2_2_0	ike_proposal_ms_2_0_0

Table 8: Phase 1 Options for OCSP Configuration Example *(Continued)*

Option	Peer A	Peer B
Certificate	local7_neg	local7_moji
Policy	ike_policy	ike_policy
Gateway address	10.0.1.2	192.0.2.0
Remote identity	fqdn company.net	fqdn company.net
Local identity	fqdn company.net	fqdn company.net
External interface	ge-1/3/0	ge-1/3/0
Version	1	1

Table 9 on page 331 shows the Phase 2 options used in this example.

Table 9: Phase 2 Options for OCSP Configuration Example

Option	Peer A	Peer B
IPsec proposal	ipsec_proposal_ms_2_2_0	ipsec_proposal_ms_2_0_0
Protocol	esp	esp
Authentication algorithm	hmac-sha1-96	hmac-sha1-96
Encryption algorithm	3des-cbc	3des-cbc
Lifetime seconds	2000	2000
IPsec policy	ipsec_policy_ms_2_2_0	ipsec_policy_ms_2_0_0

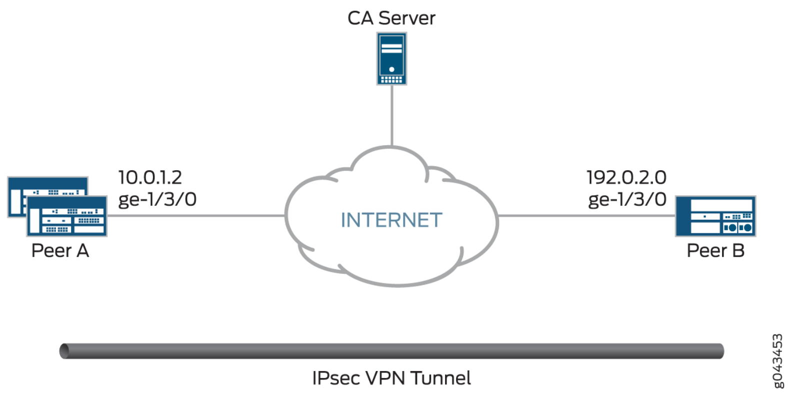
Table 9: Phase 2 Options for OCSP Configuration Example *(Continued)*

Option	Peer A	Peer B
PFC keys	group2	group2
Proposal	ipsec_proposal_ms_2_2_0	ipsec_proposal_ms_2_0_0
VPN	test_vpn	test_vpn
Policy	ipsec_policy	ipsec_policy
Establish tunnels	-	immediately

Topology

Figure 26 on page 332 shows the peer devices that are configured in this example.

Figure 26: OCSP Configuration Example



Configuration

IN THIS SECTION

Configuring Peer A | 333

- Configuring Peer B | 338

Configuring Peer A

CLI Quick Configuration

To quickly configure VPN peer A to use OCSP, 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 interfaces ge-1/3/0 unit 0 family inet address 10.0.1.2
set interfaces ms-2/2/0 unit 0 family inet
set interfaces ms-2/2/0 unit 1 family inet
set interfaces ms-2/2/0 unit 1 family inet6
set interfaces ms-2/2/0 unit 1 service-domain inside
set interfaces ms-2/2/0 unit 2 family inet
set interfaces ms-2/2/0 unit 2 family inet6
set interfaces ms-2/2/0 unit 2 service-domain outside
set security pki ca-profile Root ca-identity Root
set security pki ca-profile Root enrollment url http://10.1.1.1:8080/scep/Root/
set security pki ca-profile Root revocation-check ocsp url http://10.157.88.56:8210/Root/
set security pki ca-profile Root revocation-check use-ocsp
set security pki ca-profile Root revocation-check ocsp disable-responder-revocation-check
set security pki ca-profile Root revocation-check ocsp connection-failure fallback-crl
set services ipsec-vpn ike proposal ike_prop authentication-method rsa-signatures
set services service-set ips_ss1 next-hop-service inside-service-interface ms-2/2/0.1
set services service-set ips_ss1 next-hop-service outside-service-interface ms-2/2/0.2
set services service-set ips_ss1 ipsec-vpn-options local-gateway 10.0.1.2
set services service-set ips_ss1 ipsec-vpn-rules vpn_rule_ms_2_2_01
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 from source-address 203.0.113.0/24
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 from destination-address
198.51.100.0/24
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then remote-gateway 192.0.2.0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then dynamic ike-policy
ike_policy_ms_2_2_0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then dynamic ipsec-policy
ipsec_policy_ms_2_2_0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 match-direction input
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 protocol esp
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 authentication-algorithm hmac-
sha1-96
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 lifetime-seconds 2000
set services ipsec-vpn ipsec policy ipsec_policy_ms_2_2_0 proposals ipsec_proposal_ms_2_2_0
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 dh-group group2
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 lifetime-seconds 3000
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 mode main
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 version 1
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 proposals ike_proposal_ms_2_2_0
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 local-id fqdn company.net
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 local-certificate local7_neg
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 remote-id fqdn company.net
set services ipsec-vpn traceoptions level all
set services ipsec-vpn traceoptions flag all
set services ipsec-vpn establish-tunnels immediately
```

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* in the CLI User Guide.

To configure VPN peer A to use OCSP:

1. Configure interfaces.

```
[edit interfaces]
set interfaces ge-1/3/0 unit 0 family inet address 10.0.1.2
set interfaces ms-2/2/0 unit 0 family inet
set interfaces ms-2/2/0 unit 1 family inet
set interfaces ms-2/2/0 unit 1 family inet6
set interfaces ms-2/2/0 unit 1 service-domain inside
set interfaces ms-2/2/0 unit 2 family inet
set interfaces ms-2/2/0 unit 2 family inet6
set interfaces ms-2/2/0 unit 2 service-domain outside
```

2. Configure the CA profile.

```
[edit security pki ca-profile Root]
set security pki ca-profile Root ca-identity Root
set security pki ca-profile Root enrollment url http://10.1.1.1:8080/scep/Root/
set security pki ca-profile Root revocation-check ocsp url http://10.157.88.56:8210/Root/
set security pki ca-profile Root revocation-check use-ocsp
set security pki ca-profile Root revocation-check ocsp disable-responder-revocation-check
set security pki ca-profile Root revocation-check ocsp connection-failure fallback-crl
```

3. Configure Phase 1 options.

```
[edit services ipsec-vpn ike proposal ike_proposal_ms_2_2_0]
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 dh-group group2
set services ipsec-vpn ike proposal ike_proposal_ms_2_2_0 lifetime-seconds 3000

[edit services ipsec-vpn ike policy ike_policy_ms_2_2_0]
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 mode main
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 version 1
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 proposals ike_proposal_ms_2_2_0
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 local-id fqdn company.net
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 local-certificate local7_neg
set services ipsec-vpn ike policy ike_policy_ms_2_2_0 remote-id fqdn company.net
```

4. Configure Phase 2 options.

```
[edit services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0]
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 protocol esp
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 authentication-algorithm hmac-
sha1-96
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_2_0 lifetime-seconds 2000

[edit services ipsec-vpn ipsec policy ipsec_policy_ms_2_2_0]
set services ipsec-vpn ipsec policy ipsec_policy_ms_2_2_0 proposals ipsec_proposal_ms_2_2_0

[edit services service-set ips_ss1]
set services service-set ips_ss1 next-hop-service inside-service-interface ms-2/2/0.1
set services service-set ips_ss1 next-hop-service outside-service-interface ms-2/2/0.2
set services service-set ips_ss1 ipsec-vpn-options local-gateway 10.0.1.2
set services service-set ips_ss1 ipsec-vpn-rules vpn_rule_ms_2_2_01

[edit services ipsec-vpn rule vpn_rule_ms_2_2_01]
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 from source-address 203.0.113.0/24
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 from destination-address
198.51.100.0/24
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then remote-gateway 192.0.2.0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then dynamic ike-policy
ike_policy_ms_2_2_0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 term term11 then dynamic ipsec-policy
ipsec_policy_ms_2_2_0
set services ipsec-vpn rule vpn_rule_ms_2_2_01 match-direction input
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show security pki ca-profile Root`, `show services ipsec-vpn ike`, and `show services ipsec-vpn ipsec` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show interfaces
ge-1/3/0 {
    unit 0 {
        family inet {
            address 10.0.1.2/24;
        }
    }
}
ms-2/2/0 {
    unit 0 {
        family inet;
    }
    unit 1 {
        family inet;
            family inet6;
        service-domain inside;
    }
    unit 2 {
        family inet;
            family inet6;
        service-domain inside;
    }
}
[edit]
user@host# show security pki ca-profile Root
ca-identity Root;
enrollment {
    url http://10.1.1.1:8080/scep/Root/;
}
revocation-check {
    ocsp {
        url http://10.157.88.56:8210/Root/;
        disable-responder-revocation-check;
        connection-failure fallback-crl;
    }
    use-ocsp;
}
[edit]
user@host# show services ipsec-vpn ike
proposal ike_proposal_ms_2_2_0 {
    authentication-method rsa-signatures;
    dh-group group2;
    lifetime-seconds 3000;
}
policy ike_policy_ms_2_2_0 {
    mode main;
    version 1;
    proposals ike_proposal_ms_2_2_0;
    local-id fqdn company.net;
    local-certificate local7_neg;
    remote-id fqdn company.net;
}
[edit]
user@host# show services ipsec-vpn ipsec
proposal ipsec_proposal_ms_2_2_0 {
    protocol esp;
    authentication-algorithm hmac-sha1-96;
    encryption-algorithm 3des-cbc;
    lifetime-seconds 2000;
}
policy ipsec_policy_ms_2_2_0 {
    proposals ipsec_proposal_ms_2_2_0;
}
```

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

Configuring Peer B

CLI Quick Configuration

To quickly configure VPN peer B to use OCSP, 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 interfaces ge-1/3/0 unit 0 family inet address 192.0.2.0/24
set interfaces ms-2/0/0 unit 0 family inet
set interfaces ms-2/0/0 unit 1 family inet
set interfaces ms-2/0/0 unit 1 family inet6
set interfaces ms-2/0/0 unit 1 service-domain inside
set interfaces ms-2/0/0 unit 2 family inet
set interfaces ms-2/0/0 unit 2 family inet6
set interfaces ms-2/0/0 unit 2 service-domain outside
set security pki ca-profile Root ca-identity Root
set security pki ca-profile Root enrollment url http://10.1.1.1:8080/scep/Root/
set security pki ca-profile Root revocation-check ocsp url http://10.157.88.56:8210/Root/
set security pki ca-profile Root revocation-check use-ocsp
set security pki ca-profile Root revocation-check ocsp disable-responder-revocation-check
set security pki ca-profile Root revocation-check ocsp connection-failure fallback-crl
set services service-set ips_ss1 next-hop-service inside-service-interface ms-2/0/0.1
set services service-set ips_ss1 next-hop-service outside-service-interface ms-2/0/0.2
set services service-set ips_ss1 ipsec-vpn-options local-gateway 192.0.2.0
set services service-set ips_ss1 ipsec-vpn-rules vpn_rule_ms_2_0_01
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 from source-address 203.0.113.0/24
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 from destination-address
198.51.100.0/24
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then remote-gateway 10.0.1.2
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then dynamic ike-policy
ike_policy_ms_2_0_0
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then dynamic ipsec-policy
ipsec_policy_ms_2_0_0
set services ipsec-vpn rule vpn_rule_ms_2_0_01 match-direction input
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 protocol esp
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 authentication-algorithm hmac-
sha1-96
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 lifetime-seconds 2000
set services ipsec-vpn ipsec policy ipsec_policy_ms_2_0_0 proposals ipsec_proposal_ms_2_0_0
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 dh-group group2
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 lifetime-seconds 3000
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 mode main
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 version 1
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 proposals ike_proposal_ms_2_0_0
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 local-id fqdn company.net
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 local-certificate local7_moji
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 remote-id fqdn company.net
set services ipsec-vpn traceoptions level all
set services ipsec-vpn traceoptions flag all
```

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* in the CLI User Guide.

To configure VPN peer B to use OCSP:

1. Configure interfaces.

   ```
   [edit interfaces]
   set interfaces ge-1/3/0 unit 0 family inet address 192.0.2.0/24
   set interfaces ms-2/0/0 unit 0 family inet
   set interfaces ms-2/0/0 unit 1 family inet
   set interfaces ms-2/0/0 unit 1 family inet6
   set interfaces ms-2/0/0 unit 1 service-domain inside
   set interfaces ms-2/0/0 unit 2 family inet
   set interfaces ms-2/0/0 unit 2 family inet6
   set interfaces ms-2/0/0 unit 2 service-domain outside
   ```

2. Configure the CA profile.

   ```
   [edit security pki ca-profile Root]
   set security pki ca-profile Root ca-identity Root
   set security pki ca-profile Root enrollment url http://10.1.1.1:8080/scep/Root/
   ```

```
set security pki ca-profile Root revocation-check ocsp url http://10.157.88.56:8210/Root/
set security pki ca-profile Root revocation-check use-ocsp
set security pki ca-profile Root revocation-check ocsp disable-responder-revocation-check
set security pki ca-profile Root revocation-check ocsp connection-failure fallback-crl
```

3. Configure Phase 1 options.

```
[edit services ipsec-vpn ike proposal ike_proposal_ms_2_0_0]
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 authentication-method rsa-signatures
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 dh-group group2
set services ipsec-vpn ike proposal ike_proposal_ms_2_0_0 lifetime-seconds 3000

[edit services ipsec-vpn ike policy ike_policy_ms_2_0_0]
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 mode main
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 version 1
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 proposals ike_proposal_ms_2_0_0
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 local-id fqdn company.net
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 local-certificate local7_moji
set services ipsec-vpn ike policy ike_policy_ms_2_0_0 remote-id fqdn company.net
```

4. Configure Phase 2 options.

```
[edit services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0]
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 protocol esp
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 authentication-algorithm hmac-
sha1-96
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 encryption-algorithm 3des-cbc
set services ipsec-vpn ipsec proposal ipsec_proposal_ms_2_0_0 lifetime-seconds 2000

[edit services ipsec-vpn ipsec policy ipsec_policy_ms_2_0_0]
set services ipsec-vpn ipsec policy ipsec_policy_ms_2_0_0 proposals ipsec_proposal_ms_2_0_0

[edit services service-set ips_ss1]
set services service-set ips_ss1 next-hop-service inside-service-interface ms-2/0/0.1
set services service-set ips_ss1 next-hop-service outside-service-interface ms-2/0/0.2
set services service-set ips_ss1 ipsec-vpn-options local-gateway 192.0.2.0
set services service-set ips_ss1 ipsec-vpn-rules vpn_rule_ms_2_0_01

[edit services ipsec-vpn rule vpn_rule_ms_2_0_01]
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 from source-address 203.0.113.0/24
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 from destination-address
```

```
198.51.100.0/24
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then remote-gateway 10.0.1.2
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then dynamic ike-policy
ike_policy_ms_2_0_0
set services ipsec-vpn rule vpn_rule_ms_2_0_01 term term11 then dynamic ipsec-policy
ipsec_policy_ms_2_0_0
set services ipsec-vpn rule vpn_rule_ms_2_0_01 match-direction input
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show security pki ca-profile Root`, `show services ipsec-vpn ike`, and `show services ipsec-vpn ipsec` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show interfaces
ge-1/3/0 {
    unit 0 {
        family inet {
            address 192.0.2.0/24;
        }
    }
}
ms-2/0/0 {
    unit 0 {
        family inet;
    }
    unit 1 {
        family inet;
            family inet6;
        service-domain inside;
    }
    unit 2 {
        family inet;
            family inet6;
        service-domain inside;
    }
}
[edit]
user@host# show security pki ca-profile Root
ca-identity Root;
```

```
enrollment {
    url http://10.1.1.1:8080/scep/Root/;
}
revocation-check {
    ocsp {
        url http://10.157.88.56:8210/Root/;
        disable-responder-revocation-check;
        connection-failure fallback-crl;
    }
    use-ocsp;
}
[edit]
user@host# show services ipsec-vpn ike
proposal ike_proposal_ms_2_0_0 {
    authentication-method rsa-signatures;
    dh-group group2;
    lifetime-seconds 3000;
}
policy ike_policy_ms_2_0_0 {
    mode main;
    version 1;
    proposals ike_proposal_ms_2_0_0;
    local-id fqdn company.net;
    local-certificate local7_moji;
    remote-id fqdn company.net;
}
[edit]
user@host# show services ipsec-vpn ipsec
proposal ipsec_proposal_ms_2_0_0 {
    protocol esp;
    authentication-algorithm hmac-sha1-96;
    encryption-algorithm 3des-cbc;
    lifetime-seconds 2000;
}
policy ipsec_policy_ms_2_0_0 {
    proposals ipsec_proposal_ms_2_0_0;
}
```

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

Verification

IN THIS SECTION

- Verifying CA Certificates | 343
- Verifying Local Certificates | 345
- Verifying IKE Phase 1 Status | 346
- Verifying IPsec Phase 2 Status | 348

Confirm that the configuration is working properly.

Verifying CA Certificates

Purpose

Verify the validity of a CA certificate on each peer device.

Action

From operational mode, enter the `show security pki ca-certificate ca-profile Root` or `show security pki ca-certificate ca-profile Root detail` command.

```
user@host> show security pki ca-certificate ca-profile Root
Certificate identifier: Root
  Issued to: Root, Issued by: C = US, O = Juniper, CN = Root
  Validity:
    Not before: 07- 3-2015 10:54 UTC
    Not after: 07- 1-2020 10:54 UTC
  Public key algorithm: rsaEncryption(2048 bits)

user@host> show security pki ca-certificate ca-profile Root detail
Certificate identifier: Root
  Certificate version: 3
  Serial number: 0000a17f
  Issuer:
    Organization: Juniper, Country: US, Common name: Root
  Subject:
    Organization: Juniper, Country: US, Common name: Root
```

```
Subject string:
  C=US, O=Juniper, CN=Root
Validity:
  Not before: 07- 3-2015 10:54 UTC
  Not after: 07- 1-2020 10:54 UTC
Public key algorithm: rsaEncryption(2048 bits)
  30:82:01:0a:02:82:01:01:00:c6:38:e9:03:69:5e:45:d8:a3:ea:3d
  2e:e3:b8:3f:f0:5b:39:f0:b7:35:64:ed:60:a0:ba:89:28:63:29:e7
  27:82:47:c4:f6:41:53:c8:97:d7:1e:3c:ca:f0:a0:b9:09:0e:3d:f8
  76:5b:10:6f:b5:f8:ef:c5:e8:48:b9:fe:46:a3:c6:ba:b5:05:de:2d
  91:ce:20:12:8f:55:3c:a6:a4:99:bb:91:cf:05:5c:89:d3:a7:dc:a4
  d1:46:f2:dc:36:f3:f0:b5:fd:1d:18:f2:e6:33:d3:38:bb:44:8a:19
  ad:e0:b1:1a:15:c3:56:07:f9:2d:f6:19:f7:cd:80:cf:61:de:58:b8
  a3:f5:e0:d1:a3:3a:19:99:80:b0:63:03:1f:25:05:cc:b2:0c:cd:18
  ef:37:37:46:91:20:04:bc:a3:4a:44:a9:85:3b:50:33:76:45:d9:ba
  26:3a:3b:0d:ff:82:40:36:64:4e:ea:6a:d8:9b:06:ff:3f:e2:c4:a6
  76:ee:8b:58:56:a6:09:d3:4e:08:b0:64:60:75:f3:e2:06:91:64:73
  d2:78:e9:7a:cb:8c:57:0e:d1:9a:6d:3a:4a:9e:5b:d9:e4:a2:ef:31
  5d:2b:2b:53:ab:a1:ad:45:49:fd:a5:e0:8b:4e:0b:71:52:ca:6b:fa
  8b:0e:2c:7c:7b:02:03:01:00:01
Signature algorithm: sha1WithRSAEncryption
Distribution CRL:
  http://10.1.1.1:8080/crl-as-der/currentcrl-45.crl?id=45
Authority Information Access OCSP:
  http://10.1.1.1:8090/Root/
Use for key: CRL signing, Certificate signing, Key encipherment, Digital signature
Fingerprint:
  ed:ce:ec:13:1a:d2:ab:0a:76:e5:26:6d:2c:29:5d:49:90:57:f9:41 (sha1)
  af:87:07:69:f0:3e:f7:c6:b8:2c:f8:df:0b:ae:b0:28 (md5)
```

NOTE: In this example, IP addresses are used in the URLs in the CA profile configuration. If IP addresses are not used with CA-issued certificates or CA certificates, DNS must be configured in the device's configuration. DNS must be able to resolve the host in the distribution CRL and in the CA URL in the CA profile configuration. Additionally, you must have network reachability to the same host to receive revocation checks.

Meaning

The output shows the details and validity of CA certificate on each peer as follows:

- `C`—Country.

- `O`—Organization.
- `CN`—Common name.
- `Not before`—Begin date of validity.
- `Not after`—End date of validity.

Verifying Local Certificates

Purpose

Verify the validity of a local certificate on each peer device.

Action

From operational mode, enter the `show security pki local-certificate certificate-id localcert1 detail` command.

```
user@host> show security pki local-certificate certificate-id local7_neg detail

Certificate identifier: local7_neg
  Certificate version: 3
  Serial number: 0007d964
  Issuer:
    Organization: juniper, Country: us, Common name: Subca2
  Subject:
    Organization: juniper, Organizational unit: marketing, State: california, Locality:
sunnyvale, Common name: local, Domain component: juniper
  Subject string:
    DC=juniper, CN=local, OU=marketing, O=juniper, L=sunnyvale, ST=california, C=us
  Alternate subject: "test@company.net", company.net, 10.0.0.2
  Validity:
    Not before: 04- 5-2016 03:30 UTC
    Not after: 07- 1-2020 10:54 UTC
  Public key algorithm: rsaEncryption(1024 bits)
    30:81:89:02:81:81:00:b9:44:42:0e:26:5a:46:8e:a7:9c:b9:15:a5
    f1:38:e4:59:59:9d:84:75:ee:7a:64:ca:0a:a7:68:3b:2b:0c:dc:a8
    de:60:df:07:80:23:58:7d:56:dd:4f:50:de:a4:57:f1:a0:df:a9:7a
    6c:3d:e0:6d:7a:cf:ef:af:95:1b:12:7a:c4:54:61:12:db:65:0c:f9
    25:40:2d:01:71:21:8a:fc:fc:f6:9d:db:5a:63:ca:1a:92:2b:a3:98
    f6:6b:e4:23:67:53:92:6a:5e:ad:ae:d7:82:ab:32:c1:60:6f:01:14
    fd:46:bd:3f:b3:6b:fd:e6:41:de:6d:94:0d:6f:ad:02:03:01:00:01
  Signature algorithm: sha256WithRSAEncryption
  Distribution CRL:
    http://10.1.1.1:8080/crl-as-der/currentcrl-1925.crl?id=1925
  Authority Information Access OCSP:
    http://10.204.128.120:8090/Subca2/
  Fingerprint:
    69:00:fe:e1:81:37:ab:54:27:81:ce:57:11:a1:f2:d8:00:e7:e6:c7 (sha1)
    1e:27:93:a1:96:eb:28:0c:dc:f3:50:20:bb:eb:ed:57 (md5)
  Auto-re-enrollment:
    Status: Disabled
    Next trigger time: Timer not started
```

Meaning

The output shows the details and validity of a local certificate on each peer as follows:

- `DC`—Domain component.
- `CN`—Common name.
- `OU`—Organizational unit.
- `O`—Organization.
- `L`—Locality
- `ST`—State.
- `C`—Country.
- `Not before`—Begin date of validity.
- `Not after`—End date of validity.

Verifying IKE Phase 1 Status

Purpose

Verify the IKE Phase 1 status on each peer device.

Action

From operational mode, enter the `show services ipsec-vpn ike security-associations` command.

```
user@host> show services ipsec-vpn ike security-associations

Remote Address  State         Initiator cookie  Responder cookie  Exchange type
192.0.2.0        Matured       63b3445edda507fb  2715ee5895ed244d  Main
```

From operational mode, enter the `show services ipsec-vpn ike security-associations detail` command.

```
user@host> show services ipsec-vpn ike security-associations detail

IKE peer 192.0.2.0
  Role: Initiator, State: Matured
  Initiator cookie: 63b3445edda507fb, Responder cookie: 2715ee5895ed244d
  Exchange type: Main, Authentication method: RSA-signatures
  Local: 10.0.1.2, Remote: 192.0.2.0
  Lifetime: Expires in 788 seconds
  Algorithms:
   Authentication        : hmac-sha1-96
   Encryption            : 3des-cbc
   Pseudo random function: hmac-sha1
   Diffie-Hellman group  : 2
  Traffic statistics:
   Input  bytes  :                 3100
   Output bytes  :                 4196
   Input  packets:                    7
   Output packets:                    9
  Flags: IKE SA created
  IPSec security associations: 4 created, 4 deleted
```

Meaning

The `flags` field in the output shows that, IKE security association is created.

Verifying IPsec Phase 2 Status

Purpose

Verify the IPsec Phase 2 status on each peer device.

Action

From operational mode, enter the `show services ipsec-vpn ipsec security-associations` command.

```
user@host> show services ipsec-vpn ipsec security-associations

Service set: ips_ss1, IKE Routing-instance: default

  Rule: vpn_rule_ms_2_2_01, Term: term11, Tunnel index: 1
  Local gateway: 10.0.1.2, Remote gateway: 192.0.2.0
  IPSec inside interface: ms-2/2/0.1, Tunnel MTU: 1500
  UDP encapsulate: Disabled, UDP Destination port: 0
    Direction SPI         AUX-SPI     Mode       Type     Protocol
    inbound   2151932129  0           tunnel     dynamic  ESP
    outbound  4169263669  0           tunnel     dynamic  ESP
```

From operational mode, enter the `show services ipsec-vpn ipsec security-associations detail` command.

```
user@host> show services ipsec-vpn ipsec security-associations detail

Service set: ips_ss1, IKE Routing-instance: default

  Rule: vpn_rule_ms_2_2_01, Term: term11, Tunnel index: 1
  Local gateway: 10.0.1.2, Remote gateway: 192.0.2.0
  IPSec inside interface: ms-2/2/0.1, Tunnel MTU: 1500
  UDP encapsulate: Disabled, UDP Destination port: 0
  Local identity: ipv4_subnet(any:0,[0..7]=80.0.0.0/16)
  Remote identity: ipv4_subnet(any:0,[0..7]=30.0.0.0/16)

    Direction: inbound, SPI: 3029124496, AUX-SPI: 0
    Mode: tunnel, Type: dynamic, State: Installed
    Protocol: ESP, Authentication: hmac-sha1-96, Encryption: 3des-cbc
    Soft lifetime: Expires in 840 seconds
    Hard lifetime: Expires in 1273 seconds
    Anti-replay service: Enabled, Replay window size: 4096
    Copy ToS: Disabled, ToS value: 0
    Copy TTL: Disabled, TTL value: 64

    Direction: outbound, SPI: 4046774180, AUX-SPI: 0
    Mode: tunnel, Type: dynamic, State: Installed
    Protocol: ESP, Authentication: hmac-sha1-96, Encryption: 3des-cbc
    Soft lifetime: Expires in 840 seconds
    Hard lifetime: Expires in 1273 seconds
    Anti-replay service: Enabled, Replay window size: 4096
    Copy ToS: Disabled, ToS value: 0
    Copy TTL: Disabled, TTL value: 64
```

Meaning

The output shows the ipsec security associations details.

RELATED DOCUMENTATION

Online Certificate Status Protocol and Certificate Revocation Lists | 326

5

PART

Configuring Layer 2 Circuits

- Overview | 351
- Layer 2 Circuits Configuration Overview | 353
- Configuring Class of Service with Layer 2 Circuits | 377
- Configuring Pseudowire Redundancy for Layer 2 Circuits | 382
- Configuring Load Balancing for Layer 2 Circuits | 423
- Configuring Protection Features for Layer 2 Circuits | 424
- Monitoring Layer 2 Circuits with BFD | 474
- Troubleshooting Layer 2 Circuits | 489

CHAPTER 20

Overview

IN THIS CHAPTER

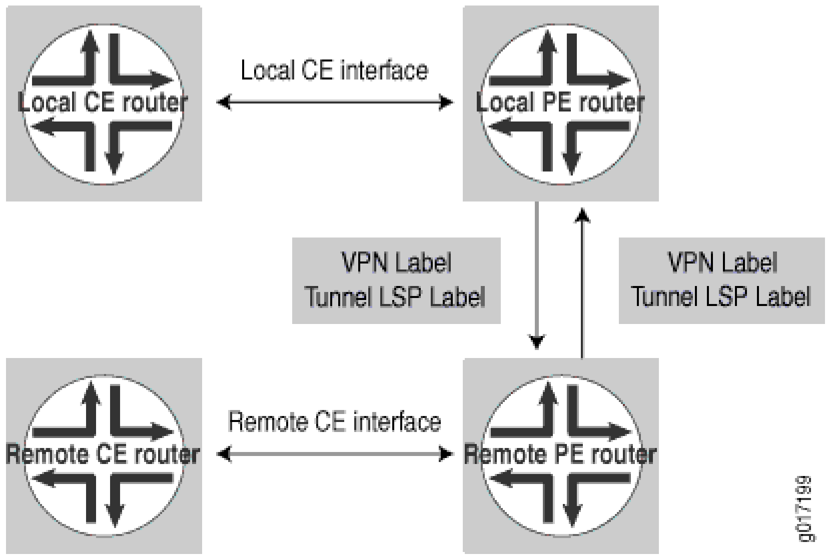
- Layer 2 Circuit Overview | **351**

Layer 2 Circuit Overview

A Layer 2 circuit is a point-to-point Layer 2 connection transported using Multiprotocol Label Switching (MPLS) or other tunneling technology on the service provider's network. A Layer 2 circuit is similar to a circuit cross-connect (CCC), except that multiple virtual circuits (VCs) are transported over a single shared label-switched path (LSP) tunnel between two provider edge (PE) routers. In contrast, each CCC requires a separate dedicated LSP.

The Junos OS implementation of Layer 2 circuits supports only the remote form of a Layer 2 circuit; that is, a connection from a local customer edge (CE) router to a remote CE router. Figure 27 on page 351 illustrates the components of a Layer 2 circuit.

Figure 27: Components of a Layer 2 Circuit



To establish a Layer 2 circuit, the Label Distribution Protocol (LDP) is used as the signaling protocol to advertise the ingress label to the remote PE routers. For this purpose, a targeted remote LDP neighbor session is established using the extended discovery mechanism described in LDP, and the session is brought up to the remote PE loopback IP address. Because LDP looks at the Layer 2 circuit configuration and initiates extended neighbor discovery for all the Layer 2 circuit neighbors (the remote PEs), no new configuration is necessary in LDP. Each Layer 2 circuit is represented by the logical interface connecting the local PE router to the local customer edge (CE) router. Note that LDP must be enabled on the lo0.0 interface for extended neighbor discovery to function correctly.

Packets are sent to remote CE routers over an egress VPN label advertised by the remote PE router, using a targeted LDP session. The VPN label is sent over an LDP LSP to the remote PE router connected to the remote CE router. Return traffic from the remote CE router destined to the local CE router is sent using an ingress VPN label advertised by the local PE router, which is also sent over the LDP LSP to the local PE router from the remote PE router.

RELATED DOCUMENTATION

Understanding Layer 3 VPNs

Layer 2 VPN Applications | **145**

Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN

Example: Interconnecting a Layer 2 Circuit with a Layer 2 Circuit | **1346**

Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN

Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN | **1335**

CHAPTER 21

Layer 2 Circuits Configuration Overview

IN THIS CHAPTER

- Configuring Static Layer 2 Circuits | **353**
- Configuring Local Interface Switching in Layer 2 Circuits | **354**
- Configuring Interfaces for Layer 2 Circuits | **357**
- Configuring Aggregated Ethernet Interfaces for Layer 2 Circuits | **367**
- Example: Configuring the Pseudowire Status TLV | **369**
- Configuring Policies for Layer 2 Circuits | **371**
- Configuring LDP for Layer 2 Circuits | **376**

Configuring Static Layer 2 Circuits

You can configure static Layer 2 circuit pseudowires. Static pseudowires are designed for networks that do not support LDP or do not have LDP enabled. You configure a static pseudowire by configuring static values for the in and out labels needed to enable a pseudowire connection. The `ignore-mtu-mismatch`, `ignore-vlan-id`, and `ignore-encapsulation-mismatch` statements are not relevant for static pseudowire configurations since the peer router cannot forward this information.

When you configure static pseudowires, you need to manually compare the encapsulation and control word of the router with the remote peer router and ensure that they match, otherwise the static pseudowire might not work.

To configure static Layer 2 circuit pseudowires, include the `static` statement:

```
static {
    incoming-label label;
    outgoing-label label;
    send-oam;
}
```

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

You can configure a static pseudowire as a standalone Layer 2 circuit or in conjunction with a redundant pseudowire. You configure the static pseudowire statement at the `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]` hierarchy level. You configure the redundant pseudowire at the `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`* `backup-neighbor` *`neighbor`*`]` hierarchy level. If you configure a static pseudowire to a neighbor and also configure a redundant pseudowire, the redundant pseudowire must also be static.

You can enable the ability to ping a static pseudowire by configuring the `send-oam` statement. This functionality applies to the backup neighbor as well. Once you have configured this statement, you can ping the static pseudowire by issuing the `ping mpls l2circuit` command.

For information about how to configure redundant pseudowires, see "Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS" on page 226.

RELATED DOCUMENTATION

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | **226**

ping mpls l2circuit

Configuring Local Interface Switching in Layer 2 Circuits

IN THIS SECTION

- Configuring the Interfaces for the Local Interface Switch | **355**
- Enabling Local Interface Switching When the MTU Does Not Match | **356**

You can configure a virtual circuit entirely on the local router, terminating the circuit on a local interface. Possible uses for this feature include being able to enable switching between Frame Relay DLCIs.

To configure a virtual circuit to terminate locally, include the `local-switching` statement:

```
local-switching {
    interface interface-name {
            description text;
            end-interface {
                interface interface-name;
                no-revert;
                protect-interface interface-name;
            }
            ignore-mtu-mismatch;
            no-revert;
            protect-interface interface-name;
        }
    }
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit]`
- `[edit logical-systems` *`logical-system-name`* `protocols l2circuit]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy level.

The following sections describe how to configure local interface switching:

Configuring the Interfaces for the Local Interface Switch

Local interface switching requires you to configure at least two interfaces:

- Starting interface—Include the `interface` statement at the `[edit protocols l2circuit local-switching]` hierarchy level.
- Ending interface—Include the `end-interface` statement at the `[edit protocols l2circuit local-switching interface` *`interface-name`*`]` hierarchy level.

You can also configure virtual circuit interface protection for each local interface:

- Protect interface for the starting interface—Include the `protect-interface` statement at the `[edit protocols l2circuit local-switching interface` *`interface-name`*`]` hierarchy level.
- Protect interface for the ending interface—Include the `protect-interface` statement at the `[edit protocols l2circuit local-switching interface` *`interface-name`* `end-interface]` hierarchy level.

For more information about how to configure protect interfaces, see "Configuring the Protect Interface" on page 357.

Typically, when the primary interface goes down, the pseudowire starts using the protect interface. By default, when the primary interface comes back online, the interface is switched-over back from the protect interface to the primary interface. To prevent the switchover back to the primary interface, unless the primary interface goes down, include the `no-revert` statement. This prevents loss of traffic during the switchover.

NOTE: If the protect interface fails, the interface is switched-over back to the primary interface, irrespective of whether or not the `no-revert` statement is included in the configuration.

You can configure the `no-revert` statement both for the starting interface and the ending interface.

```
[edit protocols l2circuit local-switching interface interface-name]
no-revert;
end-interface {
    interface interface-name;
    no-revert;
}
```

NOTE: The protect interface must be configured prior to configuring the `no-revert` statement.

Enabling Local Interface Switching When the MTU Does Not Match

You can configure a local switching interface to ignore the MTU configuration set for the associated physical interface. This enables you to bring up a circuit between two logical interfaces that are defined on physical interfaces with different MTU values.

To configure the local switching interface to ignore the MTU configured for the physical interface, include the `ignore-mtu-mismatch` statement:

```
ignore-mtu-mismatch;
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit local-switching interface` *`interface-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `protocols l2circuit local-switching interface` *`interface-name`*`]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy level.

Configuring Interfaces for Layer 2 Circuits

IN THIS SECTION

- Configuring the Address for the Neighbor of the Layer 2 Circuit | **357**
- Configuring the Neighbor Interface for the Layer 2 Circuit | **358**
- Configuring the Interface Encapsulation Type for Layer 2 Circuits | **366**
- Configuring ATM2 IQ Interfaces for Layer 2 Circuits | **366**

The following sections describe how to configure interfaces for Layer 2 circuits:

NOTE: Not all subtasks are supported on all platforms; check the CLI on your device.

Configuring the Address for the Neighbor of the Layer 2 Circuit

All the Layer 2 circuits using a particular remote PE router designated for remote CE routers are listed under the `neighbor` statement (“neighbor” designates the PE router). Each neighbor is identified by its IP address and is usually the end-point destination for the label-switched path (LSP) tunnel transporting the Layer 2 circuit.

To configure a PE router as a neighbor for a Layer 2 circuit, specify the neighbor address using the `neighbor` statement:

```
neighbor address {
    ...
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit]`
- `[edit logical-systems logical-system-name protocols l2circuit]`

Configuring the Neighbor Interface for the Layer 2 Circuit

Each Layer 2 circuit is represented by the logical interface*encapsulation* connecting the local provider edge (PE) router to the local customer edge (CE) router. This interface is tied to the Layer 2 circuit neighbor configured in "Configuring the Address for the Neighbor of the Layer 2 Circuit" on page 357.

To configure the interface for a Layer 2 circuit neighbor, include the `interface` statement:

NOTE: The commit operation fails, if the same logical interface is configured for both Layer 2 circuit and ccc connection.

NOTE: On the EX9200 switches, replace `encapsulation-type` with the *encapsulation* statement.

```
interface interface-name {
    bandwidth (bandwidth | ctnumber bandwidth);
    community community-name;
    (control-word | no-control-word);
    description text;
    encapsulation-type type;
    ignore-encapsulation-mismatch;
    ignore-mtu-mismatch;
    mtu mtu-number;
    no-revert;
    protect-interface interface-name;
    pseudowire-status-tlv;
    psn-tunnel-endpoint address;
    virtual-circuit-id identifier;
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit neighbor address]`
- `[edit logical-systems logical-system-name protocols l2circuit neighbor address]`

The following sections describe how to configure the interface for the Layer 2 circuit neighbor:

Configuring a Community for the Layer 2 Circuit

To configure a community for a Layer 2 circuit, include the `community` statement:

```
community community-name;
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit neighbor address interface interface-name]`
- `[edit logical-systems logical-system-name protocols l2circuit neighbor address interface interface-name]`

For information about how to configure a routing policy for a Layer 2 circuit, see "Configuring Policies for Layer 2 Circuits" on page 371.

Configuring the Control Word for Layer 2 Circuits

To emulate the virtual circuit (VC) encapsulation for Layer 2 circuits, a 4-byte control word is added between the Layer 2 protocol data unit (PDU) being transported and the VC label that is used for demultiplexing. For most protocols, a null control word consisting of all zeroes is sent between Layer 2 circuit neighbors.

However, individual bits are available in a control word that can carry Layer 2 protocol control information. The control information is mapped into the control word, which allows the header of a Layer 2 protocol to be stripped from the frame. The remaining data and control word can be sent over the Layer 2 circuit, and the frame can be reassembled with the proper control information at the egress point of the circuit.

The following Layer 2 protocols map Layer 2 control information into special bit fields in the control word:

- Frame Relay—The control word supports the transport of discard eligible (DE), forward explicit congestion notification (FECN), and backward explicit congestion notification (BECN) information. For configuration information, see "Configuring the Control Word for Frame Relay Interfaces" on page 360.

  NOTE: Frame Relay is not supported on the ACX Series routers.

- ATM AAL5 mode—The control word supports the transport of sequence number processing, ATM cell loss priority (CLP), and explicit forward congestion indication (EFCI) information. When you configure an AAL5 mode Layer 2 circuit, the control information is carried by default and no additional configuration is needed.

- ATM cell-relay mode—The control word supports sequence number processing only. When you configure a cell-relay mode Layer 2 circuit, the sequence number information is carried by default and no additional configuration is needed.

The Junos OS implementation of sequence number processing for ATM cell-relay mode and AAL5 mode is not the same as that described in Sec. 3.1.2 of the IETF draft *Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks*. The differences are as follows:

- A packet with a sequence number of 0 is considered as out of sequence.

- A packet that does not have the next incremental sequence number is considered out of sequence.

- When out-of-sequence packets arrive, the sequence number in the Layer 2 circuit control word increments by one and becomes the expected sequence number for the neighbor.

The following sections discuss how to configure the control word for Layer 2 circuits:

Configuring the Control Word for Frame Relay Interfaces

On interfaces with Frame Relay CCC encapsulation, you can configure Frame Relay control bit translation to support Frame Relay services over IP and MPLS backbones by using CCC, Layer 2 VPNs, and Layer 2 circuits. When you configure translation of Frame Relay control bits, the bits are mapped into the Layer 2 circuit control word and preserved across the IP or MPLS backbone.

For information about how to configure the control bits, see the Configuring Frame Relay Control Bit Translation.

Disabling the Control Word for Layer 2 Circuits

The Junos OS can typically determine whether a neighboring router supports the control word. However, if you want to explicitly disable its use on a specific interface, include the `no-control-word` statement:

```
no-control-word;
```

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

Configuring the Encapsulation Type for the Layer 2 Circuit Neighbor Interface

You can specify the Layer 2 circuit encapsulation type for the interface receiving traffic from a Layer 2 circuit neighbor. The encapsulation type is carried in the LDP-signaling messages exchanged between Layer 2 circuit neighbors when pseudowires are created. The encapsulation type you configure for each Layer 2 circuit neighbor varies depending on the type of networking equipment or the type of Layer 2 protocol you have deployed in your network. If you do not specify an encapsulation type for the Layer 2 circuit, the encapsulation of the CE device interface is used by default.

Specify the encapsulation type for the Layer 2 circuit neighbor interface by including the `encapsulation-type` statement:

```
encapsulation-type (atm-aal5 | atm-cell | atm-cell-port-mode | atm-cell-vc-mode | atm-cell-vp-mode
| cesop | cisco-hdlc | ethernet | ethernet-vlan | frame-relay | frame-relay-port-mode |
interworking | ppp | satop-e1 | satop-e3 | satop-t1 | satop-t3);
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]`

Enabling the Layer 2 Circuit When the Encapsulation Does Not Match

You can configure the Junos OS to allow a Layer 2 circuit to be established even though the encapsulation configured on the CE device interface does not match the encapsulation configured on the Layer 2 circuit interface by including the `ignore-encapsulation-mismatch` statement. You can configure the `ignore-encapsulation-mismatch` statement for the connection to the remote connection by including the statement at the `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]` hierarchy level or for the local connection by including this statement at the `[edit protocols l2circuit local-switching interface` *`interface-name`*`]` hierarchy level.

```
ignore-encapsulation-mismatch;
```

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

Configuring the MTU Advertised for a Layer 2 Circuit

By default, the MTU used to advertise a Layer 2 circuit is determined by taking the interface MTU for the associated physical interface and subtracting the encapsulation overhead for sending IP packets based on the encapsulation.

However, encapsulations that support multiple logical interfaces (and multiple Layer 2 circuits) rely on the same interface MTU (since they are all associated with the same physical interface). This can prove to be a limitation for VLAN Layer 2 circuits using the same Ethernet interface or for Layer 2 circuit DLCIs using the same Frame Relay interface.

This can also affect multivendor environments. For example, if you have three PE devices supplied by different vendors and one of the devices only supports an MTU of 1500, even if the other devices support larger MTUs you must to configure the MTU as 1500 (the smallest MTU of the three PE devices).

You can explicitly configure which MTU is advertised for a Layer 2 circuit, even if the Layer 2 circuit is sharing a physical interface with other Layer 2 circuits. When you explicitly configure an MTU for a Layer 2 circuit, be aware of the following:

- An explicitly configured MTU is signaled to the remote PE device. The configured MTU is also compared to the MTU received from the remote PE device. If there is a conflict, the Layer 2 circuit is taken down.

- If you configure an MTU for an ATM cell relay interface on an ATM II PIC, the configured MTU is used to compute the cell bundle size advertised for that Layer 2 circuit, instead of the default interface MTU.

- A configured MTU is used only in the control plane. It is not enforced in the data plane. You need to ensure that the CE device for a given Layer 2 circuit uses the correct MTU for data transmission.

To configure the MTU for a Layer 2 circuit, include the `mtu` statement at the `[edit protocols l2circuit neighbor *address* interface *interface-name*]` hierarchy level.

```
mtu mtu-number;
```

Enabling the Layer 2 Circuit When the MTU Does Not Match

You can configure the Junos OS to allow a Layer 2 circuit to be established even though the MTU configured on the PE router does not match the MTU configured on the remote PE router by including the `ignore-mtu-mismatch` statement at the `[edit protocols l2circuit neighbor *address* interface *interface-name*]` hierarchy level.

Configuring the Protect Interface

You can configure a protect interface for the logical interface linking a virtual circuit to its destination, whether the destination is remote or local. A protect interface provides a backup for the protected interface in case of failure. Network traffic uses the primary interface only so long as the primary interface functions. If the primary interface fails, traffic is switched to the protect interface. The protect interface is optional.

To configure the protect interface, include the `protect-interface` statement:

```
protect-interface interface-name;
```

NOTE: The protect interface must be configured prior to configuring the `no-revert` statement.

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

For an example of how to configure a protect interface for a Layer 2 circuit, see "Example: Configuring Layer 2 Circuit Protect Interfaces" on page 447.

Configuring the Protect Interface From Switching Over to the Primary Interface

Typically, when the primary interface goes down, the pseudowire starts using the protect interface. By default, when the primary interface comes back online, the interface is switched-over back from the protect interface to the primary interface. To prevent the switchover back to the primary interface, unless the protect interface goes down, include the `no-revert` statement. This prevents loss of traffic during the switchover.

NOTE: If the protect interface fails, the interface is switched-over back to the primary interface, irrespective of whether or not the `no-revert` statement is included in the configuration.

You can configure the `no-revert` statement at the `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]` hierarchy level:

```
[edit protocols l2circuit neighbor address interface interface-name]
no-revert;
```

Configuring the Pseudowire Status TLV

The pseudowire status type length variable (TLV) is used to communicate the status of a pseudowire back and forth between two PE routers. For Layer 2 circuit configurations, you can configure the PE router to negotiate the pseudowire with its neighbor using the pseudowire status TLV. This same functionality is also available for LDP VPLS neighbor configurations. The pseudowire status TLV is configurable for each pseudowire connection and is disabled by default. The pseudowire status negotiation process assures that a PE router reverts back to the label withdraw method for pseudowire status if its remote PE router neighbor does not support the pseudowire status TLV.

Unlike the control word, a PE router's ability to support the pseudowire status TLV is communicated when the initial label mapping message is sent to its remote PE router. Once the PE router transmits its support for the pseudowire status TLV to its remote PE router, it includes the pseudowire status TLV in every label mapping message sent to the remote PE router. If you disable support for the pseudowire status TLV on the PE router, a label withdraw message is sent to the remote PE router and then a new label mapping message without the pseudowire status TLV follows.

To configure the pseudowire status TLV for the pseudowire to the neighbor PE router, include the `pseudowire-status-tlv` statement:

```
pseudowire-status-tlv;
```

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

Configuring Layer 2 Circuits over Both RSVP and LDP LSPs

You can configure two Layer 2 circuits between the same two routers, and have one Layer 2 circuit traverse an RSVP LSP and the other traverse an LDP LSP. To accomplish this, you need to configure two loopback addresses on the local router. You configure one of the loopback address for the Layer 2 circuit traversing the RSVP LSP. You configure the other loopback address to handle the Layer 2 circuit traversing the LDP LSP. For information about how to configure multiple loop back interfaces, see *Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs*.

You also need to configure a packet switched network (PSN) tunnel endpoint for one of the Layer 2 circuits. It can be either the Layer 2 circuit traversing the RSVP LSP or the one traversing the LDP LSP. The PSN tunnel endpoint address is the destination address for the LSP on the remote router.

To configure the address for the PSN tunnel endpoint, include the `psn-tunnel-endpoint` statement:

```
psn-tunnel-endpoint address;
```

You can include this statement at the following hierarchy levels:

- `[edit logical-systems` *`logical-system-name`* `protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]`
- `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]`

By default, the PSN tunnel endpoint for a Layer 2 circuit is identical to the neighbor address, which is also the same as the LDP neighbor address.

The tunnel endpoints on the remote router do not need to be loopback addresses.

Example: PSN Tunnel Endpoint

The following example illustrates how you might configure a PSN tunnel endpoint:

```
[edit protocols l2circuit]
neighbor 10.255.0.6 {
    interface t1-0/2/2.0 {
        psn-tunnel-endpoint 192.0.2.0;
        virtual-circuit-id 1;
    }
    interface t1-0/2/1.0 {
        virtual-circuit-id 10;
    }
}
```

The Layer 2 circuit configured for the `t1-0/2/2.0` interface resolves in the inet3 routing table to `192.0.2.0`. This could be either an RSVP route or a static route with an LSP next hop.

Configuring the Virtual Circuit ID

You configure a virtual circuit ID on each interface. Each virtual circuit ID uniquely identifies the Layer 2 circuit among all the Layer 2 circuits to a specific neighbor. The key to identifying a particular Layer 2 circuit on a PE router is the neighbor address and the virtual circuit ID. An LDP-FEC-to-label binding is associated with a Layer 2 circuit based on the virtual circuit ID in the FEC and the neighbor that sent this binding. The LDP-FEC-to-label binding enables the dissemination of the VPN label used for sending traffic on that Layer 2 circuit to the remote CE device.

You also configure a virtual circuit ID for each redundant pseudowire. A redundant pseudowire is identified by the backup neighbor address and the virtual circuit ID. For more information, see "Configuring Pseudowire Redundancy on the PE Router" on page 226.

To configure the virtual circuit ID, include the `virtual-circuit-id` statement:

```
virtual-circuit-id identifier;
```

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

Configuring the Interface Encapsulation Type for Layer 2 Circuits

The Layer 2 encapsulation type is carried in the LDP forwarding equivalence class (FEC). You can configure either circuit cross-connect (CCC) or translational cross-connect (TCC) encapsulation types for Layer 2 circuits. For more information, see the MPLS Applications User Guide and Junos OS Network Interfaces Library for Routing Devices.

NOTE: Some platform and FPC combinations can not pass TCC encapsulated ISO traffic. See Platforms/FPCs That Cannot Forward TCC Encapsulated ISO Traffic for details.

To configure the interface encapsulation for a Layer 2 circuit, include the `encapsulation` statement:

```
encapsulation encapsulation;
```

You can include this statement at the following hierarchy levels:

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

Configuring ATM2 IQ Interfaces for Layer 2 Circuits

You can configure Asynchronous Transfer Mode 2 (ATM2) intelligent queuing (IQ) interfaces for Layer 2 circuits by using Layer 2 circuit ATM Adaptation Layer 5 (AAL5) transport mode, Layer 2 circuit ATM cell relay mode, and the Layer 2 circuit ATM trunk mode.

The configuration statements are as follows:

- `atm-l2circuit-mode aal5`
- `atm-l2circuit-mode cell`
- `atm-l2circuit-mode trunk`

For more information about these statements, see the Junos OS Administration Library. For more information about how to configure ATM2 IQ interfaces, see theJunos OS Network Interfaces Library for Routing Devices.

The Junos OS implementation of sequence number processing for Layer 2 circuit ATM cell relay mode and Layer 2 circuit AAL5 mode differs from that described in the Internet draft draft-martini-l2circuit-encap-mpls-11.txt, *Encapsulation Methods for Transport of Layer 2 Frames over MPLS Networks* (expires August 2006).

The Junos OS implementation has the following differences:

1. A packet with a sequence number of 0 is treated as out of sequence.
2. A packet that does not have the next incremental sequence number is considered out of sequence.

When out-of-sequence packets arrive, the expected sequence number for the neighbor is set to the sequence number in the Layer 2 circuit control word.

Configuring Aggregated Ethernet Interfaces for Layer 2 Circuits

SUMMARY

Configure AE interfaces for Layer 2 circuits. Link aggregation enhances your network resilience and increases bandwidth capacity.

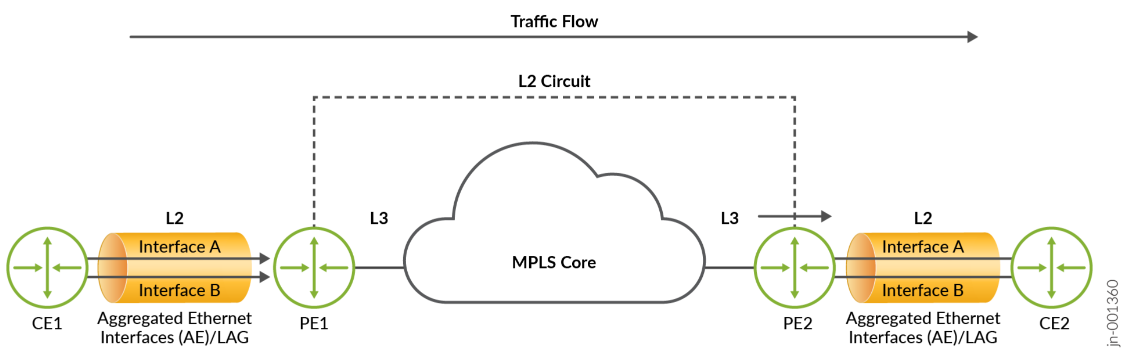
IN THIS SECTION

- Overview | 367
- Benefits of Aggregated Ethernet interfaces over Layer 2 Circuits | 368
- Configuring Aggregated Ethernet for Layer 2 Circuits | 368

Overview

Layer 2 Circuit functionality enables Layer 2 Ethernet frames to be transmitted over an IP-MPLS core. The provides seamless Layer 2 connectivity between geographically separated locations. Configuring Aggregated Ethernet (AE) interfaces for a Layer 2 Circuit topology allows you to leverage link aggregation for enhanced resilience and increased bandwidth capacity. Traffic is load-balanced across the individual links within the AE interface when being sent to the Customer Edge (CE) device. You will use existing CLI commands for configuration.

Figure 28: Aggregated Ethernet interfaces over L2 Circuit



Benefits of Aggregated Ethernet interfaces over Layer 2 Circuits

- Enhances network resilience by combining multiple physical links into a single logical link, reducing the impact of individual link failures.
- Higher data throughput and improved network performance.
- Optimized traffic distribution through load balancing.
- Simplified network management by using existing CLI commands for configuration, making it easier to set up and maintain L2 circuits on AE interfaces.

Configuring Aggregated Ethernet for Layer 2 Circuits

To configure an AE interface for an L2 Circuit, start by configuring the AE interface with an encapsulation type for your network. You can use either `ethernet-ccc` or `vlan-ccc`.

NOTE: AE interfaces cannot use `flexible-ethernet-services` with `vlan-ccc` encapsulation.

- `[edit interfaces ae`*`interface-number`*` encapsulation `*`encapsulation-type`*`]`

Finally, configure the circuit in the L2 Circuit protocols hierarchy. You will use the neighbor's IP address, the AE interface, and the virtual circuit ID:

- `[set protocols l2circuit neighbor `*`neighbor-ip-address`*` interface ae`*`interface-number virtual-circuit-number`*` virtual-circuit-id `*`virtual-circuit-id`*`]`

RELATED DOCUMENTATION

Aggregated Ethernet Interfaces

Configuring Interfaces for Layer 2 Circuits | 357

encapsulation (interfaces)

Example: Configuring the Pseudowire Status TLV

IN THIS SECTION

- Requirements | 369
- Overview | 369
- Configuration | 369

Requirements

The following is a list of the hardware and software requirements for this configuration.

- One ACX Series Universal Metro router
- Junos OS Release 12.2 or later

Overview

The configuration shown here is the base configuration of a pseudowire with pseudowire-status-tlv enabled. The pseudowire-status-tlv is used to communicate the status of a pseudowire between PE routers.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 370
- Configuring the Pseudowire Status TLV | 370
- Results | 371

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them in 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:

```
edit protocols l2circuit
set neighbor 10.255.64.26
set neighbor 10.255.64.26 interface xe-0/0/0
set neighbor 10.255.64.26 interface xe-0/0/0 pseudowire-status-tlv
set neighbor 10.255.64.26 interface xe-0/0/0 virtual-circuit-id 1024
```

Configuring the Pseudowire Status TLV

Step-by-Step Procedure

1. Navigate to the [edit protocols l2circuit] hierarchy level to configure Layer 2 circuits over MPLS.

   ```
   [edit]
   user@host# edit protocols l2circuit
   ```

2. Set the address for the neighbor provider edge router;, this example uses a fictitious address, **10.255.64.26**.

   ```
   [edit protocols l2circuit]
   user@host# set neighbor 10.255.64.26
   ```

3. Specify the name of the interface forming the Layer 2 circuit; this example uses **xe-0/0/0**.

   ```
   [edit protocols l2circuit]
   user@host# set neighbor 10.255.64.26 interface xe-0/0/0
   ```

4. Enter the **pseudowire-status-tlv** statement.

   ```
   [edit protocols l2circuit]
   user@host# set neighbor 10.255.64.26 interface xe-0/0/0 pseudowire-status-tlv
   ```

NOTE: You need to configure the **virtual-circuit-id** statement in order for **pseudowire-status-tlv** to work.

5. Set the **virtual-circuit-id** statement to identify the pseudowire as regular or redundant. The identifier value can range from 1 through 4,294,967,295.

```
[edit protocols l2circuit]
user@host# set neighbor 10.255.64.26 interface xe-0/0/0 virtual-circuit-id 1024
```

6. Check your configuration by entering the **show** command.

Results

```
[edit protocols l2circuit]
user@host# show
neighbor 10.255.64.26 {
    interface xe-0-0-0 {
        virtual-circuit-id 1024;
        pseudowire-status-tlv;
    }
}
```

RELATED DOCUMENTATION

Pseudowire Overview for ACX Series Universal Metro Routers

Configuring the Pseudowire Status TLV

Configuring Policies for Layer 2 Circuits

IN THIS SECTION

- Configuring the Layer 2 Circuit Community | 372
- Configuring the Policy Statement for the Layer 2 Circuit Community | 373

● Verifying the Layer 2 Circuit Policy Configuration | 375

You can configure Junos routing policies to control the flow of packets over Layer 2 circuits. This capability allows you to provide different level of service over a set of equal-cost Layer 2 circuits. For example, you can configure a circuit for high-priority traffic, a circuit for average-priority traffic, and a circuit for low-priority traffic. By configuring Layer 2 circuit policies, you can ensure that higher-value traffic has a greater likelihood of reaching its destination.

The following sections explain how to configure Layer 2 circuit policies:

Configuring the Layer 2 Circuit Community

To configure a community for Layer 2 circuits, include the `community` statement.

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

You can include this statement at the following hierarchy levels:

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

name identifies the community or communities.

community-ids identifies the type of community or extended community:

- A normal community uses the following community ID format:

  as-number:*community-value*

  as-number is the autonomous system (AS) number of the community member.

  community-value is the identifier of the community member. It can be a number from 0 through 65,535.

- An extended community uses the following community ID format:

  type:*administrator*:*assigned-number*

  type is the type of target community. The target community identifies the route's destination.

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

assigned-number identifies the local provider.

You also need to configure the community for the Layer 2 circuit interface; see "Configuring a Community for the Layer 2 Circuit" on page 357.

Configuring the Policy Statement for the Layer 2 Circuit Community

To configure a policy to send community traffic over a specific LSP, include the `policy-statement` statement:

```
policy-statement policy-name {
    term term-name {
        from community community-name;
        then {
            install-nexthop (except | lsp lsp-name | lsp-regex lsp-regular-expression);
            accept;
        }
    }
}
```

You can include this statement at the following hierarchy levels:

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

To prevent the installation of any matching next hops, include the `install-nexthop` statement with the `except` option:

```
install-nexthop except;
```

You can include this statement at the following hierarchy levels:

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

To assign traffic from a community to a specific LSP, include the `install-nexthop` statement with the `lsp` *`lsp-name`* option and the `accept` statement:

```
install-nexthop lsp lsp-name;
accept;
```

You can include these statements at the following hierarchy levels:

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

You can also use a regular expression to select an LSP from a set of similarly named LSPs for the `install-nexthop` statement. To configure a regular expression, include the `install-nexthop` statement with the `lsp-regex` option and the `accept` statement:

```
install-nexthop lsp-regex lsp-regular-expression;
accept;
```

You can include these statements at the following hierarchy levels:

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

Example: Configuring a Policy for a Layer 2 Circuit Community

The following example illustrates how you might configure a regular expression in a Layer 2 circuit policy. You create three LSPs to handle gold-tier traffic from a Layer 2 circuit. The LSPs are named `alpha-gold`, `beta-gold`, and `delta-gold`. You then include the `install-nexthop` statement with the `lsp-regex` option with the LSP regular expression `.*-gold` at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level:

```
[edit policy-options]
policy-statement gold-traffic {
    term to-gold-LSPs {
        from community gold;
        then {
            install-nexthop lsp-regex .*-gold;
            accept;
        }
    }
}
```

The community gold Layer 2 circuits can now use any of the -gold LSPs. Given equal utilization across the three -gold LSPs, LSP selection is made at random.

You need to apply the policy to the forwarding table. To apply a policy to the forwarding table, configure the export statement at the [edit routing-options forwarding-table] hierarchy level:

```
[edit routing-options forwarding-table]
export policy-name;
```

Verifying the Layer 2 Circuit Policy Configuration

To verify that you have configured a policy for the Layer 2 circuit, issue the show route table mpls detail command. It should display the community for ingress routes that corresponds to the Layer 2 circuits, as shown by the following example:

```
user@host> show route table mpls detail
so-1/0/1.0 (1 entry, 1 announced)
*L2VPN  Preference: 7
Next hop: via so-1/0/0.0 weight 1, selected
Label-switched-path to-community-gold
Label operation: Push 100000 Offset: -4
Next hop: via so-1/0/0.0 weight 1
Label-switched-path to-community-silver
Label operation: Push 100000 Offset: -4
Protocol next hop: 10.255.245.45
Push 100000 Offset: -4
Indirect next hop: 85333f0 314
State: <Active Int>
Local AS:   100
Age: 22
Task: Common L2 VC
Announcement bits (2): 0-KRT 1-Common L2 VC
AS path: I
Communities: 100:1
```

For more information about how to configure routing policies, see Routing Policies, Firewall Filters, and Traffic Policers User Guide.

Configuring LDP for Layer 2 Circuits

Use LDP as the signaling protocol to advertise ingress labels to the remote PE routers. When configured, LDP examines the Layer 2 circuit configuration and initiates extended neighbor discovery for all the Layer 2 circuit neighbors (for example, remote PEs). This process is similar to how LDP works when tunneled over RSVP. You must run LDP on the `lo0.0` interface for extended neighbor discovery to function correctly.

For detailed information about how to configure LDP, see the MPLS Applications User Guide.

CHAPTER 22

Configuring Class of Service with Layer 2 Circuits

IN THIS CHAPTER

- Layer 2 Circuit Bandwidth Accounting and Call Admission Control | 377
- Configuring Bandwidth Allocation and Call Admission Control in Layer 2 Circuits | 380

Layer 2 Circuit Bandwidth Accounting and Call Admission Control

IN THIS SECTION

- Bandwidth Accounting and Call Admission Control Overview | 377
- Selecting an LSP Based on the Bandwidth Constraint | 378
- LSP Path Protection and CAC | 379
- Layer 2 Circuits Trunk Mode | 380

The sections that follow discuss Layer 2 circuit bandwidth accounting and call admission control (CAC):

Bandwidth Accounting and Call Admission Control Overview

Some network environments require that a certain level of service be guaranteed across the entire length of a path transiting a service provider's network. For Layer 2 circuits transiting an MPLS core network, a customer requirement might be to assure that guarantees for bandwidth and *class of service* (CoS) be maintained across the core network. For example, an Asynchronous Transfer Mode (ATM) circuit can provide service guarantees for each traffic class. A Layer 2 circuit configured to transport that ATM circuit across the network could be expected to provide the same service guarantees.

Providing this type of service guarantee requires the following:

- The LSPs in the MPLS core network must be able to provide service guarantees for bandwidth, rerouting, and route failures. You accomplish these guarantees by configuring multiclass LSPs. For more information about multiclass LSPs, see Configuring Multiclass LSPs.
- The service guarantee must be maintained across the entire length of the link as it transits the service provider's network. Different Layer 2 circuits could have different bandwidth requirements. However, many Layer 2 circuits could be transported over the same E-LSP in the MPLS core network.
- CAC ensures that the LSP has sufficient bandwidth to accommodate the Layer 2 circuit. If there is not enough bandwidth over a particular LSP, the Layer 2 circuit is prevented from using that LSP.

Selecting an LSP Based on the Bandwidth Constraint

CAC of Layer 2 circuits is based on the bandwidth constraint. You must configure this constraint for each Layer 2 circuit interface. If there is a bandwidth constraint configured for a Layer 2 circuit, CAC bases the final selection of which LSP-forwarding next hop to use on the following:

- If multiple LSPs meet the bandwidth requirements, the first LSP found that can satisfy the bandwidth requirements for the Layer 2 circuit is selected.
- If there is more than one next hop mapped to the same LSP, then all the next hops that map to that LSP and pass CAC constraints are installed. This allows the Layer 2 circuit routes to restore themselves quickly in case of failure.
- The available bandwidth on the selected LSP is decremented by the bandwidth required for each Layer 2 circuit. Similarly, when the Layer 2 circuit route is changed or deleted (for example, when the route is disassociated from that particular LSP), the bandwidth on the corresponding LSP is incremented.
- There are no priorities among different Layer 2 circuits competing for the same LSP next hop in the core network.
- When an LSP's bandwidth changes, the Layer 2 circuits using that LSP repeat the CAC process again.

  If the LSP bandwidth increases, some Layer 2 circuits that were not established might now successfully resolve over the LSP. Similarly, if the bandwidth of the LSP decreases, some Layer 2 circuits that were previously up might now be declared down because of insufficient bandwidth on the LSP.
- When no LSP is found to meet the bandwidth requirements of the Layer 2 circuit, it is considered to be a CAC failure, and an error is reported.

LSP Path Protection and CAC

CAC can take into account LSPs that have been configured with an MPLS path protection feature, such as secondary paths, fast reroute, or node and link protection. CAC can consider the bandwidth available on these auxiliary links and can accept the backup connection as valid if the main connection fails. However, there are limitations on how the path protection feature must be configured to prevent CAC from taking down the Layer 2 circuit when the LSP it is using is switched to a backup route.

For more information about MPLS path protection features, see the MPLS and Traffic Protection.

The sections that follow discuss the path protection features that can be used in conjunction with CAC and how they must be configured:

Secondary Paths and CAC

The following describes the ways in which secondary paths would interact with Layer 2 circuit CAC:

- If an LSP is configured with both primary and secondary paths, if the paths have the same bandwidth, and if this bandwidth is enough to accommodate the Layer 2 circuit, the Layer 2 circuit route installs both next hops in the forwarding table.

  CAC allows the Layer 2 circuit to be switched to the secondary path if the primary path fails.

- If the LSP has primary and secondary paths configured with different bandwidths, each path must run through CAC independently. If the active path for that LSP passes CAC constraints successfully, then that next hop is installed and the corresponding LSP is selected to transport the Layer 2 circuit traffic. The LSP's secondary paths are then checked for CAC, and installed if there is sufficient bandwidth.

  However, if the active path for the LSP fails to meet the CAC constraints, then that LSP is not selected and the system looks for a different LSP to transport the Layer 2 circuit.

  For example, an LSP has an active primary path with 30 megabits of bandwidth and a secondary path with 10 megabits of bandwidth. The Layer 2 circuit requires 15 megabits of bandwidth. The secondary path fails CAC, and only the next hop corresponding to the primary path is installed for the Layer 2 circuit route. The path protection originally provided by the secondary path is no longer available.

Fast Reroute and CAC

No CAC is done for fast reroute detours. However, as long as the protected path satisfies the CAC bandwidth constraints, the detour next hop is also selected and installed.

Link and Node Protection and CAC

You can configure CAC on Layer 2 circuit-based LSPs with bandwidth constraints and also enable link and node protection. However, if the primary LSP fails, CAC might not be applied to the bypass LSP, meaning the bypass LSP might not meet the bandwidth constraint for the Layer 2 circuit. To minimize the risk of loosing traffic, the Layer 2 circuit continues to use the non-CAC bypass LSP while an attempt is made to establish a new Layer 2 circuit route over an LSP that does support CAC.

Layer 2 Circuits Trunk Mode

Using Layer 2 circuit trunk mode, you can configure Layer 2 circuits to carry ATM trunks, providing a way to link ATM switches over an MPLS core network.

Layer 2 circuit trunk mode allows you to configure the following CoS features:

- CoS queues in Layer 2 circuit trunk mode—For ATM2 IQ interfaces, you can configure ATM CoS queues for Layer 2 circuit trunk mode.
- Layer 2 circuit trunk mode scheduling—For ATM2 IQ interfaces configured to use Layer 2 circuit trunk mode, you can share a scheduler among 32 trunks on an ATM port.
- Two early packet discard (EPD) thresholds per queue—For ATM2 IQ interfaces configured to use Layer 2 circuit trunk mode, you can set two EPD thresholds that depend on the packet-loss priorities (PLPs) of the packets.

For a detailed overview and configuration documentation, see the ATM Interfaces User Guide for Routing Devices and Class of Service User Guide (Routers and EX9200 Switches).

RELATED DOCUMENTATION

MPLS and Traffic Protection

Configuring Bandwidth Allocation and Call Admission Control in Layer 2 Circuits

You can configure bandwidth allocation and call admission control (CAC) on Layer 2 circuits. This feature is available for RSVP-signaled LSPs traversing an MPLS network.

When you enable bandwidth allocation on a Layer 2 circuit, attempts to establish an RSVP-signaled LSP are preceded by a check of the available bandwidth on the network. This check is the CAC. The available bandwidth is compared to the bandwidth requested by the LSP. If there is insufficient bandwidth, the

Layer 2 circuit is not established and an error message is generated. To apply CAC to a Layer 2 circuit, a bandwidth constraint must be configured.

You can specify the bandwidth for a Layer 2 circuit without configuring a bandwidth for each class type (queue). To specify the bandwidth allocation for a Layer 2 circuit, include the `bandwidth` statement:

```
bandwidth bandwidth;
```

Specify the bandwidth in bits per second.

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit neighbor address interface interface-name]`
- `[edit logical-systems logical-system-name protocols l2circuit neighbor address interface interface-name]`

Alternatively, you can configure the bandwidth for each class type on a Layer 2 circuit. If you use this type of configuration, you cannot simultaneously configure the nonclass type of bandwidth configuration for the Layer 2 circuit (the commit operation fails).

To configure the bandwidth for each class type on an Layer 2 circuit, include the `bandwidth` statement:

```
bandwidth {
    ct0 bandwidth;
    ct1 bandwidth;
    ct2 bandwidth;
    ct3 bandwidth;
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit neighbor address interface interface-name]`
- `[edit logical-systems logical-system-name protocols l2circuit neighbor address interface interface-name]`

Specify the bandwidth for each class type in bits per second. It is not necessary to specify a bandwidth for all four class types.

CHAPTER 23

Configuring Pseudowire Redundancy for Layer 2 Circuits

IN THIS CHAPTER

- Understanding Pseudowire Redundancy Mobile Backhaul Scenarios | **382**
- Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario | **388**
- Extension of Pseudowire Redundancy Condition Logic to Pseudowire Service Logical Interface Overview | **419**

Understanding Pseudowire Redundancy Mobile Backhaul Scenarios

IN THIS SECTION

- Sample Topology | **383**
- Benefits of Pseudowire Redundancy Mobile Backhaul | **383**
- Layer 2 Virtual Circuit Status TLV Extension | **384**
- How It Works | **385**

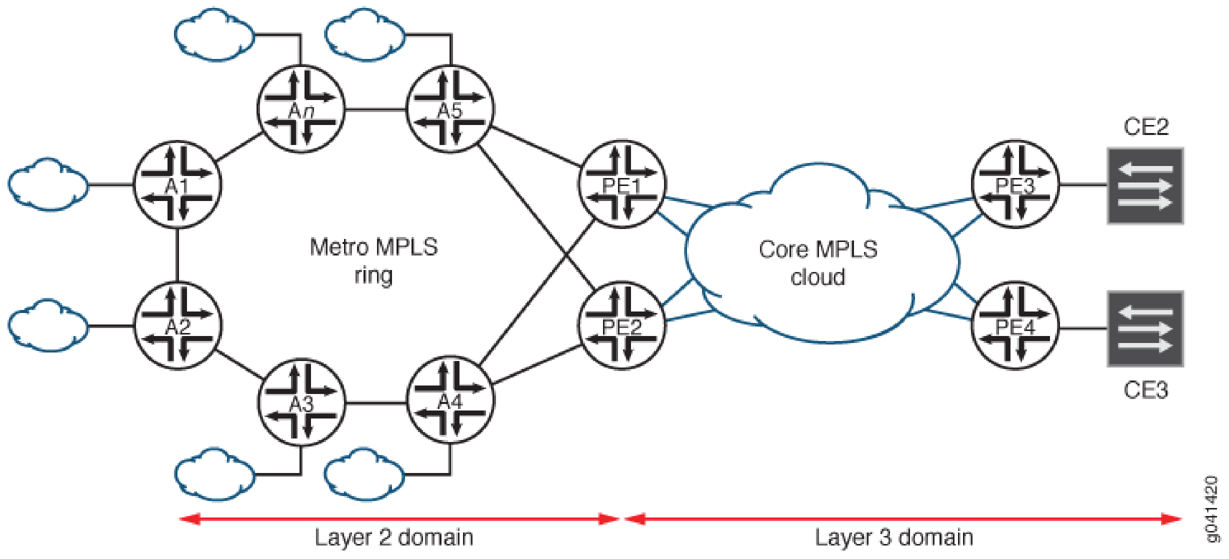
With the rising demand for mobile broadband services, telecommunication providers are seeing a sharp increase in bandwidth requirements. To keep pace with demand, operators are deploying packet-based backhaul networks that offer increased capacity at a lower cost, while providing the necessary service reliability and quality of experience that users expect.

Most of the legacy backhaul infrastructure has been traditionally built over PDH microwave, TDM T1/E1, or ATM-over-DSL links. Service providers have traditionally added subsequent TDM links to their base stations when needed to deal with bandwidth constraint scenarios. This expansion model has proven to be inefficient for the unprecedented traffic demands required by 3G and Long Term Evolution (LTE) services. As a direct consequence, operators are gradually migrating to an Ethernet-based higher capacity infrastructure in the backhaul portion of 3G and LTE topologies. Modern base stations now provide Ethernet backhaul connectivity, allowing pseudowire technologies to transport end-user content to the desired destination. As part of this Ethernet transition, service providers are increasingly demanding better resiliency mechanisms to cover the existence gap with those features provided by previous legacy technologies. With that goal in mind, Junos OS provides efficient pseudowire redundancy capabilities to those topologies where Layer 2 and Layer 3 segments are interconnected.

Sample Topology

Figure 29 on page 383 shows a sample topology.

Figure 29: Pseudowire Redundancy Mobile Backhaul Sample Topology



Benefits of Pseudowire Redundancy Mobile Backhaul

Junos OS pseudowire redundancy capabilities are as follows:

- Redundant loop-free paths to interconnect Layer 2 and Layer 3 domains.
- Layer 2 and Layer 3 domains are synchronized with regard to the elected data path.
- Traffic disruption is minimal for the following possible scenarios:
  - Access link failures
  - Node failures
  - Control-plane failures

- Traffic interruption is minimal after the failure's restoration is completed.

Layer 2 Virtual Circuit Status TLV Extension

The pseudowire status TLV is used to communicate the status of a pseudowire between provider edge (PE) routers. To avoid potential primary-path discrepancies, there must be a mechanism that allows all network elements to be synchronized with respect to the primary path over which traffic needs to be sent. With this goal in mind, the status TLV is extended to address this requirement.

NOTE: The pseudowire status TLV is not supported by ACX5000 line of routers.

By having the active and standby states being defined by the access routers, Junos OS mitigates potential primary path collisions, as there is a unique network element dictating the preferable forwarding path to be elected. As an added value, this allows network operators to switch forwarding paths on demand, which is quite useful for troubleshooting and network maintenance purposes.

The active and standby states are communicated to the aggregation routers by making use of an additional pseudowire state flag.

Table 10 on page 384 includes a list of the pseudowire state flags.

Table 10: Pseudowire Status Code for the Pseudowire Status TLV

Flag	Code
L2CKT_PW_STATUS_PW_FWD	0x00000000
L2CKT_PW_STATUS_PW_NOT_FWD	0x00000001
L2CKT_PW_STATUS_AC_RX_FAULT	0x00000002
L2CKT_PW_STATUS_AC_TX_FAULT	0x00000004
L2CKT_PW_STATUS_PSN_RX_FAULT	0x00000008
L2CKT_PW_STATUS_PSN_TX_FAULT	0x00000010

Table 10: Pseudowire Status Code for the Pseudowire Status TLV *(Continued)*

Flag	Code
L2CKT_PW_STATUS_PW_FWD_STDBY	0x00000020 Indicates the standby state.
L2CKT_PW_STATUS_SWITCH_OVER	0x00000040

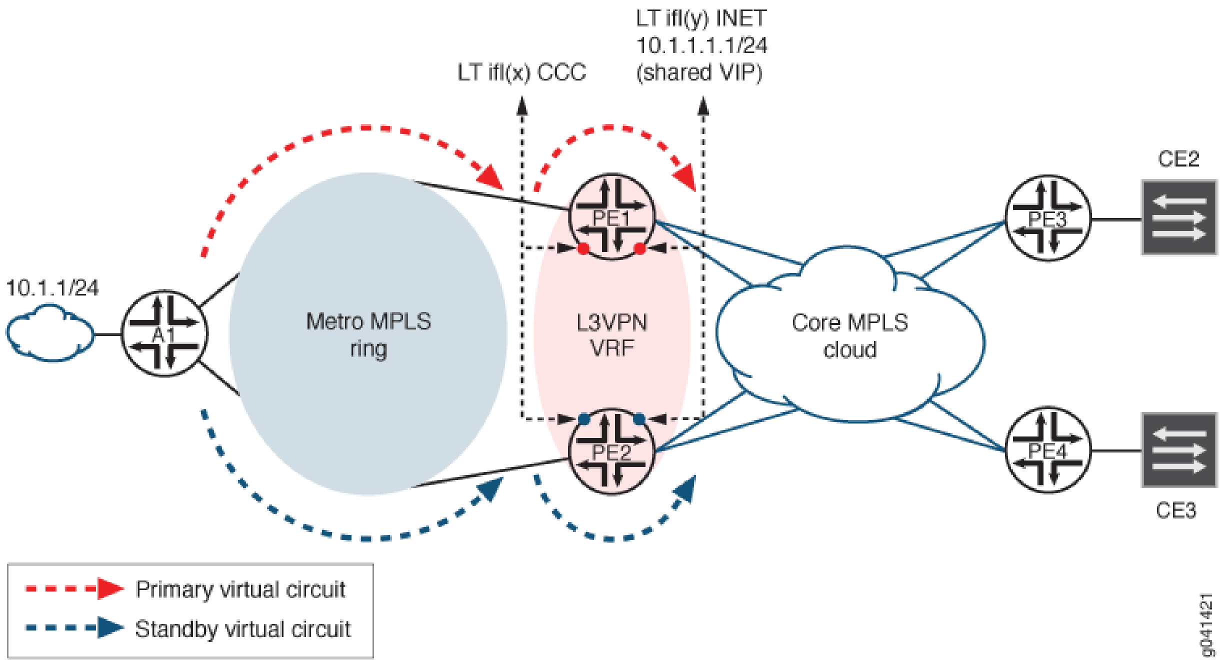
In multichassis LAG (MC-LAG)-based scenarios, this same PW_FWD_STDBY flag is used to advertise to remote PE devices which attachment circuit (AC) is being used as the active one. Upon arrival of this flag, the receiving PE device drops any pseudowire built toward the router originating this state. As we can see, this behavior denotes a slightly different semantic for the PW_FWD_STDBY flag. As a consequence, you can configure the `hot-standby-vc-on` statement to control whether the pseudowire must be constructed upon arrival of the PW_FWD_STDBY flag (in the hot-standby pseudowire scenario), or simply destroy it (in the MC-LAG scenario).

How It Works

The solution uses logical tunnel (lt-) paired interfaces for stitching the Layer 2 and Layer 3 domains.

Figure 30 on page 386 shows a diagram depicting how pseudowire redundancy in a mobile backhaul scenario works.

Figure 30: Pseudowire Redundancy Mobile Backhaul Solution



A Layer 2 pseudowire terminates on one of the logical tunnel interfaces (x), defined with the circuit cross-connect (CCC) address family configured. A Layer 3 VPN (RFC 2547) terminates the second logical tunnel interface (y), defined with the IPv4 (inet) address family. Logical tunnel interface (x) and (y) are paired. Layer 2 pseudowires established between each access router and its corresponding aggregation PE devices terminate on the logical tunnel interface defined within each PE device. This logical tunnel interface is used to establish a Layer 2 virtual circuit (VC) toward the remote end. In consequence, the CCC address family needs to be configured on it. The same applies to the remote end, where an equivalent interface needs to be defined with CCC capabilities.

This CCC logical tunnel interface created in the aggregation PE devices is paired with a second logical tunnel interface on which the INET address family is enabled. This second logical tunnel interface is configured within the context of an RFC 2547 Layer 3 VPN.

Within the scope of this document, we refer to the CCC and INET logical tunnel interfaces as LT(x) and LT(y), respectively.

The Junos OS routing protocol process (rpd) enables the stitching required to interconnect the Layer 2 VC ending in LT(x) and the associated LT(y).

In the aggregation PE routers, the routing process builds a pseudowire toward access routers, and this happens regardless of the active or standby state of the pseudowire. The same occurs in access routers, where the control and forwarding state is preestablished in both the Routing Engine and the Packet Forwarding Engine to mitigate traffic disruption during convergence periods.

An attachment circuit (AC) is a physical or virtual circuit (VC) that attaches a CE device to a PE device. Local preference is used to provide better information than the multiple exit discriminator (MED) value provides for a packet's path selection. You can configure the local preference attribute so that it has a higher value for prefixes received from a router that provides a desired path than prefixes received from a router that provides a less desirable path. The higher the value, the more preferred the route. The local preference attribute is the metric most often used in practice to express preferences for one set of paths over another.

If the Layer 2 circuit is primary, the corresponding PE device advertises the AC's subnet with the higher local preference. All aggregation PE devices initially advertise the AC's subnet with the same local preference. You can configure a routing policy to allow a higher local preference value to be advertised if the Layer 2 VC is active.

If a pseudowire is down, LT(x) is tagged with the CCC_Down flag. When this happens, the corresponding PE device withdraws the AC subnet that was initially advertised. The LT(y) address is shared between the aggregation PE devices as a virtual instance port (VIP). No VRRP hello messages are exchanged. Both PE devices assume the primary role.

Both primary and standby Layer 2 VCs are kept open to reduce traffic disruption in backup-to-primary transitions. The `hot-standby-vc-on` *configuration statement* allows manual activation.

Resiliency in the Layer 2 domain is provided through plain pseudowire redundancy for back-to-back connections. For other topologies, pseudowire virtual circuit connectivity verification (VCCV) is used.

Resiliency in the Layer 3 domain is provided by MPLS fast reroute and end-to-end service restoration. A restoration timer prevents having VCs in the secondary path from being switched back to the primary path immediately after the primary PE device is restored.

Access routers can indicate to the aggregation routers which Layer 2 VC is considered to be active. Upon arrival at LT(x) of a status TLV message communicating a standby state, the routing process decreases the BGP's local preference value of the direct subnet represented by the LT(y) IPv4 address. At this point, BGP proceeds to advertise this local preference change to the rest of the members within the Layer 3 domain, which will then reelect the designated forwarder PE device by relying on BGP's path selection mechanisms.

A similar behavior occurs upon arrival of a status TLV message indicating a Layer 2 VC active state. In this case, the receiving PE device changes the local preference corresponding to the LT(y)'s subnet. The value to be used to either decrease or increase the subnet's local preference value is manually configured using a policy.

RELATED DOCUMENTATION

Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario | **388**

Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario

IN THIS SECTION

- Requirements | 388
- Overview | 388
- Configuration | 389
- Verification | 414

This example shows how to configure pseudowire redundancy where Layer 2 and Layer 3 segments are interconnected in a mobile backhaul scenario.

Requirements

This example can be configured using the following hardware and software components:

- Junos OS Release 13.2 or later
- ACX5000 line of routers as the access (A) routers
- MX Series 5G Universal Routing Platforms or M Series Multiservice Edge Routers for the Provider Edge (PE) Routers
- PTX Series Packet Transport Routers acting as transit label-switched routers
- T Series Core Routers for the Core Routers

NOTE: The PE routers could also be T Series Core Routers but that is not typical. Depending on your scaling requirements, the core routers could also be MX Series 5G Universal Routing Platforms or M Series Multiservice Edge Routers. The Customer Edge (CE) devices could be other routers or switches from Juniper Networks or another vendor.

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

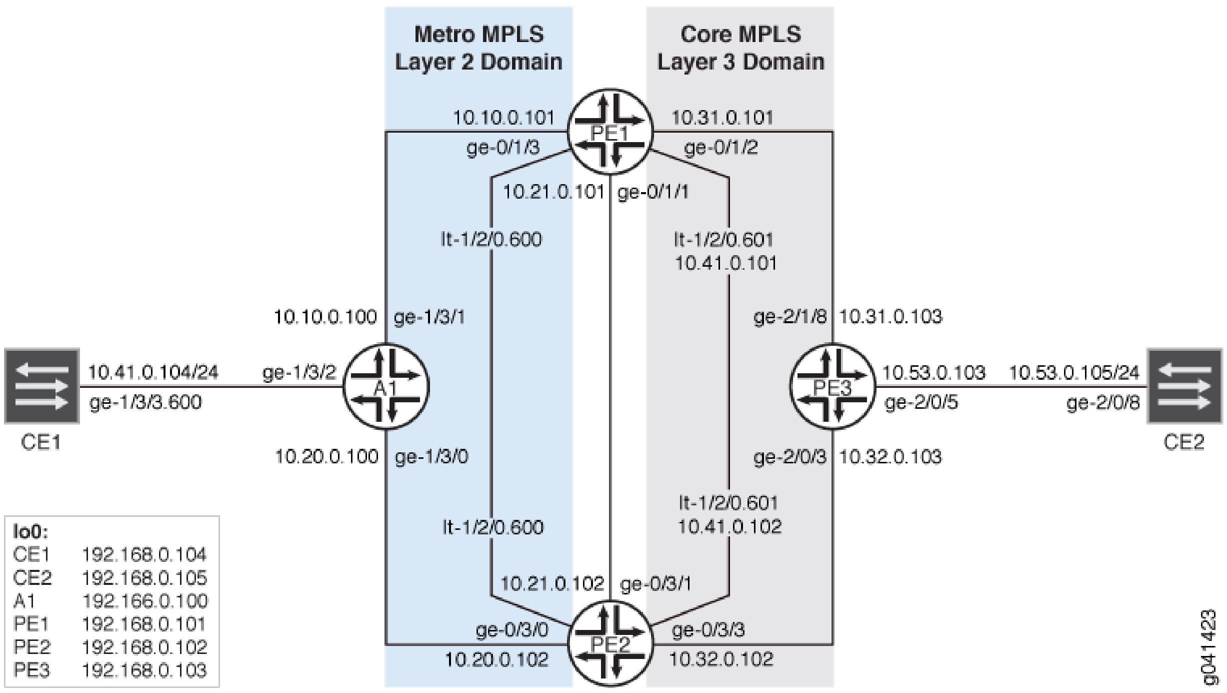
Overview

Device CE1 is a simple edge router with an IPv4 interface and a static route pointing to the PE devices. Device A1 establishes two virtual circuits (VCs) toward Device PE1 and Device PE2 by making use of the `hot-standby` statement. Device PE1 and Device PE2 terminate these VCs and enforce a policy condition over the logical tunnel IPv4 subnet. Device PE3 performs as a Layer 3 VPN provider edge device by having an IPv4 interface in a Layer 3 VPN shared with Device PE1 and Device PE2.

"CLI Quick Configuration" on page 390 shows the configuration for all of the devices in Figure 31 on page 389.

The section "Step-by-Step Procedure" on page 397 describes the steps on Device A1 and Device PE1.

Figure 31: Pseudowire Redundancy in a Mobile Backhaul Example Topology



Configuration

IN THIS SECTION

- Procedure | 390

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 CE1

```
set interfaces ge-1/3/3 vlan-tagging
set interfaces ge-1/3/3 unit 600 vlan-id 600
set interfaces ge-1/3/3 unit 600 family inet address 10.41.0.104/24
set interfaces lo0 unit 0 family inet address 192.168.0.104/32 primary
set protocols ospf area 0.0.0.0 interface ge-1/3/3.600
set protocols ospf area 0.0.0.0 interface lo0.0
set routing-options static route 192.168.0.0/8 next-hop 10.41.0.1
set routing-options static route 10.53.0.0/16 next-hop 10.41.0.1
set routing-options router-id 192.168.0.104
```

Device A1

```
set interfaces ge-1/3/0 unit 0 family inet address 10.20.0.100/24
set interfaces ge-1/3/0 unit 0 family iso
set interfaces ge-1/3/0 unit 0 family mpls
set interfaces ge-1/3/1 unit 0 family inet address 10.10.0.100/24
set interfaces ge-1/3/1 unit 0 family iso
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/2 vlan-tagging
set interfaces ge-1/3/2 encapsulation vlan-ccc
set interfaces ge-1/3/2 unit 600 encapsulation vlan-ccc
set interfaces ge-1/3/2 unit 600 vlan-id 600
set interfaces ge-1/3/2 unit 600 family ccc
set interfaces lo0 unit 0 family inet address 192.168.0.100/32 primary
set interfaces lo0 unit 0 family iso address 49.0002.0192.0168.0100.00
set routing-options router-id 192.168.0.100
set routing-options autonomous-system 64510
set routing-options forwarding-table export pplb
set protocols rsvp interface ge-1/3/0.0
set protocols rsvp interface ge-1/3/1.0
set protocols rsvp interface lo0.0
set protocols mpls interface ge-1/3/0.0
set protocols mpls interface ge-1/3/1.0
set protocols isis interface ge-1/3/0.0
set protocols isis interface ge-1/3/1.0
set protocols isis interface lo0.0
set protocols ldp interface ge-1/3/0.0
set protocols ldp interface ge-1/3/1.0
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600 virtual-circuit-id 1
set protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600 pseudowire-status-tlv
set protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600 revert-time 10 maximum 60
set protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600 backup-neighbor
192.168.0.102 virtual-circuit-id 2
set protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600 backup-neighbor
192.168.0.102 hot-standby
set policy-options policy-statement pplb then load-balance per-packet
```

Device PE1

```
set interfaces ge-0/1/1 unit 0 family inet address 10.21.0.101/24
set interfaces ge-0/1/1 unit 0 family iso
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces ge-0/1/2 unit 0 family inet address 10.31.0.101/24
set interfaces ge-0/1/2 unit 0 family iso
set interfaces ge-0/1/2 unit 0 family mpls
set interfaces ge-0/1/3 unit 0 family inet address 10.10.0.101/24
set interfaces ge-0/1/3 unit 0 family iso
set interfaces ge-0/1/3 unit 0 family mpls
set interfaces lt-1/2/0 unit 600 encapsulation vlan-ccc
set interfaces lt-1/2/0 unit 600 vlan-id 600
set interfaces lt-1/2/0 unit 600 peer-unit 601
set interfaces lt-1/2/0 unit 601 encapsulation vlan
set interfaces lt-1/2/0 unit 601 vlan-id 600
set interfaces lt-1/2/0 unit 601 peer-unit 600
set interfaces lt-1/2/0 unit 601 family inet filter input icmp_inet
set interfaces lt-1/2/0 unit 601 family inet filter output icmp_inet
set interfaces lt-1/2/0 unit 601 family inet address 10.41.0.101/24 vrrp-group 0 virtual-address
10.41.0.1
set interfaces lt-1/2/0 unit 601 family inet address 10.41.0.101/24 vrrp-group 0 accept-data
set interfaces lo0 unit 0 family inet address 192.168.0.101/32 primary
set interfaces lo0 unit 0 family iso address 49.0002.0192.0168.0003.00
set interfaces lo0 unit 1 family inet address 192.168.1.101/32
set routing-options router-id 192.168.0.101
set routing-options autonomous-system 64511
set protocols rsvp interface ge-0/1/1.0
set protocols rsvp interface ge-0/1/2.0
set protocols rsvp interface ge-0/1/3.0
set protocols rsvp interface lo0.0
set protocols mpls label-switched-path to_PE3 to 192.168.0.103
set protocols mpls label-switched-path to_PE2 to 192.168.0.102
set protocols mpls interface ge-0/1/1.0
set protocols mpls interface ge-0/1/2.0
set protocols mpls interface ge-0/1/3.0
set protocols bgp local-address 192.168.0.101
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp peer-as 64511
set protocols bgp group ibgp neighbor 192.168.0.102
set protocols bgp group ibgp neighbor 192.168.0.103
set protocols isis interface ge-0/1/1.0
set protocols isis interface ge-0/1/2.0
set protocols isis interface ge-0/1/3.0
set protocols isis interface lo0.0
set protocols ldp interface ge-0/1/1.0
set protocols ldp interface ge-0/1/2.0
set protocols ldp interface ge-0/1/3.0
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 192.168.0.100 interface lt-1/2/0.600 virtual-circuit-id 1
set protocols l2circuit neighbor 192.168.0.100 interface lt-1/2/0.600 pseudowire-status-tlv hot-
standby-vc-on
set policy-options policy-statement l3vpn_export term primary from condition primary
set policy-options policy-statement l3vpn_export term primary then local-preference add 300
set policy-options policy-statement l3vpn_export term primary then community set l3vpn
set policy-options policy-statement l3vpn_export term primary then accept
set policy-options policy-statement l3vpn_export term standby from condition standby
set policy-options policy-statement l3vpn_export term standby then local-preference add 30
set policy-options policy-statement l3vpn_export term standby then community set l3vpn
set policy-options policy-statement l3vpn_export term standby then accept
set policy-options policy-statement l3vpn_export term default then community set l3vpn
set policy-options policy-statement l3vpn_export term default then accept
set policy-options policy-statement l3vpn_import term 1 from community l3vpn
set policy-options policy-statement l3vpn_import term 1 then accept
set policy-options policy-statement l3vpn_import term default then reject
set policy-options policy-statement ospf_export term 0 from community l3vpn
set policy-options policy-statement ospf_export term 0 then accept
set policy-options community l3vpn members target:64511:600
set policy-options condition primary if-route-exists address-family ccc lt-1/2/0.600
set policy-options condition primary if-route-exists address-family ccc table mpls.0
set policy-options condition primary if-route-exists address-family ccc peer-unit 601
set policy-options condition standby if-route-exists address-family ccc lt-1/2/0.600
set policy-options condition standby if-route-exists address-family ccc table mpls.0
set policy-options condition standby if-route-exists address-family ccc standby
set policy-options condition standby if-route-exists address-family ccc peer-unit 601
set firewall family inet filter icmp_inet interface-specific
set firewall family inet filter icmp_inet term 0 from source-address 10.41.0.101/32 except
set firewall family inet filter icmp_inet term 0 from source-address 10.0.0.0/8
set firewall family inet filter icmp_inet term 0 from protocol icmp
set firewall family inet filter icmp_inet term 0 then count icmp_inet
set firewall family inet filter icmp_inet term 0 then log
set firewall family inet filter icmp_inet term 0 then accept
set firewall family inet filter icmp_inet term 1 then accept
set routing-instances l3vpn instance-type vrf
set routing-instances l3vpn interface lt-1/2/0.601
set routing-instances l3vpn interface lo0.1
set routing-instances l3vpn route-distinguisher 192.168.1.101:64511
set routing-instances l3vpn vrf-import l3vpn_import
set routing-instances l3vpn vrf-export l3vpn_export
set routing-instances l3vpn vrf-table-label
set routing-instances l3vpn protocols ospf export ospf_export
set routing-instances l3vpn protocols ospf area 0.0.0.0 lt-1/2/0.601
set routing-instances l3vpn protocols ospf area 0.0.0.0 lo0.1
```

Device PE2

```
set interfaces ge-0/3/0 unit 0 family inet address 10.20.0.102/24
set interfaces ge-0/3/0 unit 0 family iso
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces ge-0/3/1 unit 0 family inet address 10.21.0.102/24
set interfaces ge-0/3/1 unit 0 family iso
set interfaces ge-0/3/1 unit 0 family mpls
set interfaces ge-0/3/3 unit 0 family inet address 10.32.0.102/24
set interfaces ge-0/3/3 unit 0 family iso
set interfaces ge-0/3/3 unit 0 family mpls
set interfaces lt-1/2/0 unit 600 encapsulation vlan-ccc
set interfaces lt-1/2/0 unit 600 vlan-id 600
set interfaces lt-1/2/0 unit 600 peer-unit 601
set interfaces lt-1/2/0 unit 601 encapsulation vlan
set interfaces lt-1/2/0 unit 601 vlan-id 600
set interfaces lt-1/2/0 unit 601 peer-unit 600
set interfaces lt-1/2/0 unit 601 family inet filter input icmp_inet
set interfaces lt-1/2/0 unit 601 family inet filter output icmp_inet
set interfaces lt-1/2/0 unit 601 family inet address 10.41.0.102/24 vrrp-group 0 virtual-address
10.41.0.1
set interfaces lt-1/2/0 unit 601 family inet address 10.41.0.102/24 vrrp-group 0 accept-data
set interfaces lo0 unit 0 family inet address 192.168.0.102/32 primary
set interfaces lo0 unit 0 family iso address 49.0002.0192.0168.0102.00
set interfaces lo0 unit 1 family inet address 192.168.1.102/32
set routing-options router-id 192.168.0.102
set routing-options autonomous-system 64511
set protocols rsvp interface ge-0/3/0.0
set protocols rsvp interface ge-0/3/1.0
set protocols rsvp interface ge-0/3/3.0
set protocols rsvp interface lo0.0
set protocols mpls label-switched-path to_PE1 to 192.168.0.101
set protocols mpls label-switched-path to_PE3 to 192.168.0.103
set protocols mpls interface ge-0/3/0.0
set protocols mpls interface ge-0/3/1.0
set protocols mpls interface ge-0/3/3.0
set protocols bgp local-address 192.168.0.102
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp peer-as 64511
set protocols bgp group ibgp neighbor 192.168.0.101
set protocols bgp group ibgp neighbor 192.168.0.103
set protocols isis interface ge-0/3/0.0
set protocols isis interface ge-0/3/1.0
set protocols isis interface ge-0/3/3.0
set protocols isis interface lo0.0
set protocols ldp interface ge-0/3/0.0
set protocols ldp interface ge-0/3/1.0
set protocols ldp interface ge-0/3/3.0
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 192.168.0.100 interface lt-1/2/0.600 virtual-circuit-id 2
set protocols l2circuit neighbor 192.168.0.100 interface lt-1/2/0.600 pseudowire-status-tlv hot-
standby-vc-on
set policy-options policy-statement l3vpn_export term primary from condition primary
set policy-options policy-statement l3vpn_export term primary then local-preference add 300
set policy-options policy-statement l3vpn_export term primary then community set l3vpn
set policy-options policy-statement l3vpn_export term primary then accept
set policy-options policy-statement l3vpn_export term standby from condition standby
set policy-options policy-statement l3vpn_export term standby then local-preference add 30
set policy-options policy-statement l3vpn_export term standby then community set l3vpn
set policy-options policy-statement l3vpn_export term standby then accept
set policy-options policy-statement l3vpn_export term default then community set l3vpn
set policy-options policy-statement l3vpn_export term default then accept
set policy-options policy-statement l3vpn_import term 1 from community l3vpn
set policy-options policy-statement l3vpn_import term 1 then accept
set policy-options policy-statement l3vpn_import term default then reject
set policy-options policy-statement ospf_export term 0 from community l3vpn
set policy-options policy-statement ospf_export term 0 then accept
set policy-options community l3vpn members target:64511:600
set policy-options condition primary if-route-exists address-family ccc lt-1/2/0.600
set policy-options condition primary if-route-exists address-family ccc table mpls.0
set policy-options condition primary if-route-exists address-family ccc peer-unit 601
set policy-options condition standby if-route-exists address-family ccc lt-1/2/0.600
set policy-options condition standby if-route-exists address-family ccc table mpls.0
set policy-options condition standby if-route-exists address-family ccc standby
set policy-options condition standby if-route-exists address-family ccc peer-unit 601
set firewall family inet filter icmp_inet interface-specific
set firewall family inet filter icmp_inet term 0 from source-address 10.41.0.102/32 except
set firewall family inet filter icmp_inet term 0 from source-address 10.0.0.0/8
set firewall family inet filter icmp_inet term 0 from protocol icmp
set firewall family inet filter icmp_inet term 0 then count icmp_inet
set firewall family inet filter icmp_inet term 0 then log
set firewall family inet filter icmp_inet term 0 then accept
set firewall family inet filter icmp_inet term 1 then accept
set routing-instances l3vpn instance-type vrf
set routing-instances l3vpn interface lt-1/2/0.601
set routing-instances l3vpn interface lo0.1
set routing-instances l3vpn route-distinguisher 192.168.1.102:64511
set routing-instances l3vpn vrf-import l3vpn_import
set routing-instances l3vpn vrf-export l3vpn_export
set routing-instances l3vpn vrf-table-label
set routing-instances l3vpn protocols ospf export ospf_export
set routing-instances l3vpn protocols ospf area 0.0.0.0 interface lt-1/2/0.601
set routing-instances l3vpn protocols ospf area 0.0.0.0 interface lo0.1
```

Device PE3

```
set interfaces ge-2/0/3 unit 0 family inet address 10.32.0.103/24
set interfaces ge-2/0/3 unit 0 family iso
set interfaces ge-2/0/3 unit 0 family mpls
set interfaces ge-2/0/5 unit 0 family inet address 10.53.0.103/24
set interfaces ge-2/0/5 unit 0 family mpls
set interfaces ge-2/1/8 unit 0 family inet address 10.31.0.103/24
set interfaces ge-2/1/8 unit 0 family iso
set interfaces ge-2/1/8 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.0.103/32 primary
set interfaces lo0 unit 0 family iso address 49.0002.0192.0168.0103.00
set interfaces lo0 unit 1 family inet address 192.168.1.103/32
set routing-options router-id 192.168.0.103
set routing-options autonomous-system 64511
set protocols rsvp interface ge-2/0/3.0
set protocols rsvp interface ge-2/1/8.0
set protocols rsvp interface lo0.0
set protocols mpls label-switched-path to_PE1 to 192.168.0.101
set protocols mpls label-switched-path to_PE2 to 192.168.0.102
set protocols mpls interface ge-2/0/3.0
set protocols mpls interface ge-2/1/8.0
set protocols bgp local-address 192.168.0.103
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp peer-as 64511
set protocols bgp group ibgp neighbor 192.168.0.101
set protocols bgp group ibgp neighbor 192.168.0.102
set protocols isis interface ge-2/0/3.0
set protocols isis interface ge-2/1/8.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/3.0
set protocols ldp interface ge-2/1/8.0
set protocols ldp interface lo0.0
set policy-options policy-statement l3vpn_ospf_export term 0 from protocol direct
set policy-options policy-statement l3vpn_ospf_export term 0 then accept
set policy-options policy-statement l3vpn_ospf_import term 0 from protocol bgp
set policy-options policy-statement l3vpn_ospf_import term 0 from community l3vpn
set policy-options policy-statement l3vpn_ospf_import term 0 then accept
set policy-options policy-statement ospf_export term 0 from community l3vpn
set policy-options policy-statement ospf_export term 0 then accept
set policy-options community l3vpn members target:64511:600
set routing-instances l3vpn instance-type vrf
set routing-instances l3vpn interface ge-2/0/5.0
set routing-instances l3vpn interface lo0.1
set routing-instances l3vpn route-distinguisher 192.168.0.103:64511
set routing-instances l3vpn vrf-target target:64511:600
set routing-instances l3vpn vrf-table-label
set routing-instances l3vpn protocols ospf export ospf_export
set routing-instances l3vpn protocols ospf area 0.0.0.0 interface ge-2/0/5.0
set routing-instances l3vpn protocols ospf area 0.0.0.0 interface lo0.1
```

Device CE2

```
set interfaces ge-2/0/8 unit 0 family inet address 10.53.0.105/24
set interfaces lo0 unit 0 family inet address 192.168.0.105/32 primary
set protocols ospf area 0.0.0.0 interface ge-2/0/8.0
set protocols ospf area 0.0.0.0 interface lo0.0
set routing-options router-id 192.168.0.105
```

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 Device A1:

1. Configure the interfaces.

   Enable MPLS on the core-facing interfaces. The ISO address family is also enabled, because IS-IS is used as the interior gateway protocol (IGP) in the provider network.

   On the customer-facing interface, you do not need to enable MPLS. On this interface, enable CCC encapsulation and address family CCC.

   ```
   [edit interfaces]
   user@A1# set ge-1/3/0 unit 0 family inet address 10.20.0.100/24
   user@A1# set ge-1/3/0 unit 0 family iso
   user@A1# set ge-1/3/0 unit 0 family mpls
   user@A1# set ge-1/3/1 unit 0 family inet address 10.10.0.100/24
   user@A1# set ge-1/3/1 unit 0 family iso
   user@A1# set ge-1/3/1 unit 0 family mpls
   user@A1# set ge-1/3/2 vlan-tagging
   user@A1# set ge-1/3/2 encapsulation vlan-ccc
   user@A1# set ge-1/3/2 unit 600 encapsulation vlan-ccc
   user@A1# set ge-1/3/2 unit 600 vlan-id 600
   user@A1# set ge-1/3/2 unit 600 family ccc
   user@A1# set lo0 unit 0 family inet address 192.168.0.100/32 primary
   user@A1# set lo0 unit 0 family iso address 49.0002.0192.0168.0100.00
   ```

2. Configure the RSVP on the core-facing interfaces and on the loopback interface.

RSVP is used in the Layer 3 domain.

```
[edit protocols rsvp]
user@A1# set interface ge-1/3/0.0
user@A1# set interface ge-1/3/1.0
user@A1# set interface lo0.0
```

3. Configure LDP on the core-facing interfaces and on the loopback interface.

   LDP is used in Layer 2 domain.

```
[edit protocols ldp]
user@A1# set interface ge-1/3/0.0
user@A1# set interface ge-1/3/1.0
user@A1# set interface lo0.0
```

4. Configure MPLS on the core-facing interfaces.

```
[edit protocols mpls]
user@A1# set interface ge-1/3/0.0
user@A1# set interface ge-1/3/1.0
```

5. Configure an interior gateway protocol, such as IS-IS or OSPF, on the core-facing interfaces and on the loopback interface.

```
[edit protocols isis]
user@A1# set interface ge-1/3/0.0
user@A1# set interface ge-1/3/1.0
user@A1# set interface lo0.0
```

6. On the interface that faces the customer edge, configure the Layer 2 circuit.

   Configure the `hot-standby` statement on those routers with both active and standby virtual circuits (VCs) (Device A1 in our topology). You must include the `pseudowire-status-tlv` statement on access routers. Without the status TLV signaling, the standby flag cannot be advertised to remote provider edge (PE) devices.

   The `revert-time` statement and the `maximum` option should also be configured on access routers. Without the `revert-time` statement, traffic of all the VCs will not be transitioned to the primary path upon completion of the restoration. If a `revert-time` delay is defined but a `maximum` delay is not, then VCs are restored immediately upon the revert timer's expiration. The maximum option allows the VCs to be restored in a scattered fashion rather than all at once.

```
[edit protocols l2circuit neighbor 192.168.0.101 interface ge-1/3/2.600]
user@A1# set virtual-circuit-id 1
user@A1# set pseudowire-status-tlv
user@A1# set revert-time 10 maximum 60
user@A1# set backup-neighbor 192.168.0.102 virtual-circuit-id 2
user@A1# set backup-neighbor 192.168.0.102 hot-standby
```

7. To have the unilist next hop get pushed to other access routers, configure per-packet load balancing.

```
[edit policy-options policy-statement pplb]
user@A1# set then load-balance per-packet
```

8. Apply the per-packet load balancing policy.

```
[edit routing-options forwarding-table]
user@A1# set export pplb
```

9. Configure the autonomous system (AS) ID and the router ID.

```
[edit routing-options]
user@A1# set router-id 192.168.0.100
user@A1# set autonomous-system 64510
```

Similarly, configure any other access devices.

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 Device PE1:

1. Configure the interfaces.

Enable MPLS on the core-facing interfaces.

```
[edit interfaces]
user@PE1# set ge-0/1/1 unit 0 family inet address 10.21.0.101/24
user@PE1# set ge-0/1/1 unit 0 family iso
user@PE1# set ge-0/1/1 unit 0 family mpls
user@PE1# set ge-0/1/2 unit 0 family inet address 10.31.0.101/24
user@PE1# set ge-0/1/2 unit 0 family iso
user@PE1# set ge-0/1/2 unit 0 family mpls
user@PE1# set ge-0/1/3 unit 0 family inet address 10.10.0.101/24
user@PE1# set ge-0/1/3 unit 0 family iso
user@PE1# set ge-0/1/3 unit 0 family mpls
user@PE1# set lo0 unit 0 family inet address 192.168.0.101/32 primary
user@PE1# set lo0 unit 0 family iso address 49.0002.0192.0168.0003.00
user@PE1# set lo0 unit 1 family inet address 192.168.1.101/32
```

2. On Device PE1 and Device PE2, which are aggregation routers, configure a pair of logical tunnel interfaces to represent LT(x) and LT(y).

   The solution uses logical tunnel (lt-) paired interfaces for stitching the Layer 2 and Layer 3 domains.

   A Layer 2 pseudowire terminates on one of the logical tunnel interfaces, LT(x), defined with the circuit cross-connect (CCC) address family. A Layer 3 VPN terminates the second logical tunnel interface, LT(y), defined with the IPv4 (inet) address family. LT(x) and LT(y) are paired.

```
[edit interfaces]
user@PE1# set lt-1/2/0 unit 600 encapsulation vlan-ccc
user@PE1# set lt-1/2/0 unit 600 vlan-id 600
user@PE1# set lt-1/2/0 unit 600 peer-unit 601
user@PE1# set lt-1/2/0 unit 601 encapsulation vlan
user@PE1# set lt-1/2/0 unit 601 vlan-id 600
user@PE1# set lt-1/2/0 unit 601 peer-unit 600
user@PE1# set lt-1/2/0 unit 601 family inet filter input icmp_inet
user@PE1# set lt-1/2/0 unit 601 family inet filter output icmp_inet
```

3. (Optional) Associate a unique VRRP address with both Device PE1 and Device PE2.

In this case, both Device PE1 and Device PE2 assume the primary state for the defined VIP IPv4 address, so no VRRP hello message are exchanged between the routers.

```
[edit interfaces lt-1/2/0 unit 601 family inet address 10.41.0.101/24]
user@PE1# set vrrp-group 0 virtual-address 10.41.0.1
user@PE1# set vrrp-group 0 accept-data
```

4. Configure IS-IS or another IGP.

```
[edit protocols isis]
user@PE1# set interface ge-0/1/1.0
user@PE1# set interface ge-0/1/2.0
user@PE1# set interface ge-0/1/3.0
user@PE1# set interface lo0.0
```

5. Configure the MPLS on the core-facing interfaces.

```
[edit protocols mpls]
user@PE1# set interface ge-0/1/1.0
user@PE1# set interface ge-0/1/2.0
user@PE1# set interface ge-0/1/3.0
```

6. Configure label-switched paths to the other PE devices.

   BGP is a policy-driven protocol, so also configure and apply any needed routing policies. For example, you might want to export static routes into BGP.

```
[edit protocols mpls]
user@PE1# set label-switched-path to_PE3 to 192.168.0.103
user@PE1# set label-switched-path to_PE2 to 192.168.0.102
```

7. Configure LDP on the core-facing interfaces and on the loopback interface.

```
[edit protocols ldp]
user@PE1# set interface ge-0/1/1.0
user@PE1# set interface ge-0/1/2.0
user@PE1# set interface ge-0/1/3.0
user@PE1# set interface lo0.0
```

8. Configure RSVP on the core-facing interfaces and on the loopback interface.

```
[edit protocols rsvp]
user@PE1# set interface ge-0/1/1.0
user@PE1# set interface ge-0/1/2.0
user@PE1# set interface ge-0/1/3.0
user@PE1# set interface lo0.0
```

9. Configure internal BGP (IBGP).

```
[edit protocols bgp]
user@PE1# set local-address 192.168.0.101
user@PE1# set group ibgp family inet-vpn any
user@PE1# set group ibgp peer-as 64511
user@PE1# set group ibgp neighbor 192.168.0.102
user@PE1# set group ibgp neighbor 192.168.0.103
```

10. Configure the Layer 2 circuit on the logical tunnel interface.

    Configure the `hot-standby-vc-on` statement if you want a hot standby pseudowire to be established upon arrival of PW_FWD_STDBY status TLV.

```
[edit protocols l2circuit neighbor 192.168.0.100 interface lt-1/2/0.600]
user@PE1# set virtual-circuit-id 1
user@PE1# set pseudowire-status-tlv hot-standby-vc-on
```

11. Define a pair of conditions to be applied to the egress policy defined within the Layer 3 VPN instance.

    In both condition `primary` and condition `standby`, the matching route corresponds to the interface lt-1/2/0.600 (y), as this is the format in which egress routes appear in routing table mpls.0 to represent any given pseudowire.

    The difference between these conditions is in the `standby` attribute. Upon arrival of the PW_FWD_STDBY status TLV to Device PE1 or Device PE2, Junos OS matches condition `standby`, and in consequence, only term `standby` within the `l3vpn` policy will be executed. On the other hand, if the PW_FWD_STDBY status TLV is not present, the policy only matches condition `primary`, which then executes term `primary` in the `l3vpn` policy. Also, for logical tunnel-based CCC services, you must specify the logical tunnel interface, LT(y), that is associated with the logical tunnel CCC interface, LT(x). (See "Understanding Pseudowire Redundancy Mobile Backhaul Scenarios" on page 382.)

Finally, for CCC-based conditions, Junos OS only allows mpls.0 as the matching routing table. For the `address` attribute, Junos OS only allows strings with a logical interface unit format (for example, lt-0/0/0.0).

```
[edit policy-options condition primary if-route-exists address-family ccc]
user@PE1# set lt-1/2/0.600
user@PE1# set table mpls.0
user@PE1# set peer-unit 601
[edit policy-options condition standby if-route-exists address-family ccc]
user@PE1# set lt-1/2/0.600
user@PE1# set table mpls.0
user@PE1# set standby
user@PE1# set peer-unit 601
```

12. Configure the Layer 3 VPN export policy.

    If the Layer 2 virtual circuit (VC) is primary, the corresponding provider edge (PE) routing device advertises the attachment circuit's (AC's) subnet with the higher local preference. All aggregation PE devices initially advertise the AC's subnet with the same local preference.

    This routing policy allows a higher local preference value to be advertised if the Layer 2 VC is active.

```
[edit policy-options policy-statement l3vpn_export]
user@PE1# set term primary from condition primary
user@PE1# set term primary then local-preference add 300
user@PE1# set term primary then community set l3vpn
user@PE1# set term primary then accept
user@PE1# set term standby from condition standby
user@PE1# set term standby then local-preference add 30
user@PE1# set term standby then community set l3vpn
user@PE1# set term standby then accept
user@PE1# set term default then community set l3vpn
user@PE1# set term default then accept
```

13. Configure the Layer 3 VPN community members.

```
[edit policy-options community l3vpn]
user@PE1# set members target:64511:600
```

14. Configure the Layer 3 VPN import policy, based on the Layer 3 VPN community.

```
[edit policy-options policy-statement l3vpn_import]
user@PE1# set term 1 from community l3vpn
user@PE1# set term 1 then accept
user@PE1# set term default then reject
```

15. Configure OSPF export policy, based on the Layer 3 VPN community.

```
[edit policy-options policy-statement ospf_export term 0]
user@PE1# set from community l3vpn
user@PE1# set then accept
```

16. (Optional) Configure a firewall filter to check the path taken by traffic.

```
[edit firewall family inet filter icmp_inet]
user@PE1# set interface-specific
user@PE1# set term 0 from source-address 10.41.0.101/32 except
user@PE1# set term 0 from source-address 10.0.0.0/8
user@PE1# set term 0 from protocol icmp
user@PE1# set term 0 then count icmp_inet
user@PE1# set term 0 then log
user@PE1# set term 0 then accept
user@PE1# set term 1 then accept
```

17. Configure the routing instance.

    This routing instance is in the Layer 2 domain where Device PE1 and Device PE2 are interconnected to the metro ring over multiaccess media (Ethernet). You must include the `vrf-table-label`' statement on Device PE1 and Device PE2 to enable advertisement of the direct subnet prefix corresponding to the logical tunnel (lt-) interface toward the Layer 3 domain.

    Device PE1 and Device PE2 use OSPF for Layer 3 VPN communication with Device CE1.

```
[edit routing-instances l3vpn]
user@PE1# set instance-type vrf
user@PE1# set interface lt-1/2/0.601
user@PE1# set interface lo0.1
user@PE1# set route-distinguisher 192.168.1.101:64511
user@PE1# set vrf-import l3vpn_import
```

```
user@PE1# set vrf-export l3vpn_export
user@PE1# set vrf-table-label
user@PE1# set protocols ospf export ospf_export
user@PE1# set protocols ospf area 0.0.0.0 interface lt-1/2/0.601
user@PE1# set protocols ospf area 0.0.0.0 interface lo0.1
```

18. Configure the autonomous system (AS) ID and router ID.

```
[edit routing-options]
user@PE1# set router-id 192.168.0.101
user@PE1# set autonomous-system 64511
```

Similarly, configure Device PE2.

Results

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

Device A1

```
user@A1#


                  show interfaces


ge-1/3/0 {
    unit 0 {
        family inet {
            address 10.20.0.100/24;
        }
        family iso;
        family mpls;
    }
}
ge-1/3/1 {
    unit 0 {
        family inet {
            address 10.10.0.100/24;
        }
        family iso;
        family mpls;
    }
}
ge-1/3/2 {
    vlan-tagging;
    encapsulation vlan-ccc;
    unit 600 {
        encapsulation vlan-ccc;
        vlan-id 600;
        family ccc;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.0.100/32 {
                primary;
            }
        }
        family iso {
            address 49.0002.0192.0168.0100.00;
        }
    }
}
```

```
user@A1#

                   show protocols



rsvp {
    interface ge-1/3/0.0;
    interface ge-1/3/1.0;
    interface lo0.0;
}
mpls {
    interface ge-1/3/0.0;
    interface ge-1/3/1.0;
}
```

```
isis {
    interface ge-1/3/0.0;
    interface ge-1/3/1.0;
    interface lo0.0;
}
ldp {
    interface ge-1/3/0.0;
    interface ge-1/3/1.0;
    interface lo0.0;
}
l2circuit {
    neighbor 192.168.0.101 {
        interface ge-1/3/2.600 {
            virtual-circuit-id 1;
            pseudowire-status-tlv;
            backup-neighbor 192.168.0.102 {
                virtual-circuit-id 2;
                hot-standby;
            }
        }
    }
}
```

```
user@A1#

                  show policy-options


policy-statement pplb {
    then {
        load-balance per-packet;
    }
}
```

```
user@A1#

                  show routing-options


autonomous-system 64510;
```

```
router-id 192.168.0.100;
forwarding-table {
    export pplb;
}
```

Device PE1

```
user@PE1#

                   show interfaces



ge-0/1/1 {
    unit 0 {
        family inet {
            address 10.21.0.101/24;
        }
        family iso;
        family mpls;
    }
}
ge-0/1/2 {
    unit 0 {
        family inet {
            address 10.31.0.101/24;
        }
        family iso;
        family mpls;
    }
}
ge-0/1/3 {
    unit 0 {
        family inet {
            address 10.10.0.101/24;
        }
        family iso;
        family mpls;
    }
}
lt-1/2/0 {
    unit 600 {
        encapsulation vlan-ccc;
```

```
            vlan-id 600;
            peer-unit 601;
        }
        unit 601 {
            encapsulation vlan;
            vlan-id 600;
            peer-unit 600;
            family inet {
                filter {
                    input icmp_inet;
                    output icmp_inet;
                }
                address 10.41.0.101/24 {
                    vrrp-group 0 {
                        virtual-address 10.41.0.1;
                        accept-data;
                    }
                }
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.0.101/32 {
                    primary;
                }
            }
            family iso {
                address 49.0002.0192.0168.0003.00;
            }
        }
        unit 1 {
            family inet {
                address 192.168.1.101/32;
            }
        }
    }
```

```
user@PE1#
```

```
                   show firewall


family inet {
    filter icmp_inet {
        interface-specific;
        term 0 {
            from {
                source-address {
                    10.41.0.101/32 except;
                    10.0.0.0/8;
                }
                protocol icmp;
            }
            then {
                count icmp_inet;
                log;
                accept;
            }
        }
        term 1 {
            then accept;
        }
    }
}
```

```
user@PE1#

                   show protocols


rsvp {
    interface ge-0/1/1.0;
    interface ge-0/1/2.0;
    interface ge-0/1/3.0;
    interface lo0.0;
}
mpls {
    label-switched-path to_PE3 {
        to 192.168.0.103;
    }
    label-switched-path to_PE2 {
        to 192.168.0.102;
    }
    interface ge-0/1/1.0;
    interface ge-0/1/2.0;
    interface ge-0/1/3.0;
}
bgp {
    local-address 192.168.0.101;
    group ibgp {
        family inet-vpn {
            any;
        }
        peer-as 64511;
        neighbor 192.168.0.102;
        neighbor 192.168.0.103;
    }
}
isis {
    interface ge-0/1/1.0;
    interface ge-0/1/2.0;
    interface ge-0/1/3.0;
    interface lo0.0;
}
ldp {
    interface ge-0/1/1.0;
    interface ge-0/1/2.0;
    interface ge-0/1/3.0;
    interface lo0.0;
}
l2circuit {
    neighbor 192.168.0.100 {
        interface lt-1/2/0.600 {
            virtual-circuit-id 1;
            pseudowire-status-tlv hot-standby-vc-on;
        }
    }
}
```

```
user@PE1#
```

```
                   show policy-options


policy-statement l3vpn_export {
    term primary {
        from condition primary;
        then {
            local-preference add 300;
            community set l3vpn;
            accept;
        }
    }
    term standby {
        from condition standby;
        then {
            local-preference add 30;
            community set l3vpn;
            accept;
        }
    }
    term default {
        then {
            community set l3vpn;
            accept;
        }
    }
}
policy-statement l3vpn_import {
    term 1 {
        from community l3vpn;
        then accept;
    }
    term default {
        then reject;
    }
}
policy-statement ospf_export {
    term 0 {
        from community l3vpn;
        then accept;
    }
}
community l3vpn members target:64511:600;
```

```
condition primary {
    if-route-exists {
        address-family {
            ccc {
                lt-1/2/0.600;
                table mpls.0;
                peer-unit 601;
            }
        }
    }
}
condition standby {
    if-route-exists {
        address-family {
            ccc {
                lt-1/2/0.600;
                table mpls.0;
                standby;
                peer-unit 601;
            }
        }
    }
}
```

```
user@PE1#

                   show routing-options


router-id 192.168.0.101;
autonomous-system 64511;
```

```
user@PE1#

                   show routing-instances


l3vpn {
    instance-type vrf;
    interface lt-1/2/0.601;
    interface lo0.1;
    route-distinguisher 192.168.1.101:64511;
    vrf-import l3vpn_import;
    vrf-export l3vpn_export;
    vrf-table-label;
    protocols {
        ospf {
            export ospf_export;
            area 0.0.0.0 {
                interface lt-1/2/0.601;
                interface lo0.1;
            }
        }
    }
}
```

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

Verification

IN THIS SECTION

- Checking Layer 2 Circuits | **414**
- Checking the Policy Conditions | **418**

Confirm that the configuration is working properly.

Checking Layer 2 Circuits

Purpose

Upon Layer 2 virtual circuit (VC) establishment, the output of the `show l2circuit connections` command shows the active and the hot-standby VC. In addition, control-plane details are shown for the hot-standby VC.

Action

From operational mode, enter the `show l2circuit connections extensive` command.

```
user@A1> show l2circuit connections extensive

Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  HS -- Hot-standby Connection
XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.168.0.101
    Interface                 Type  St     Time last up          # Up trans
    ge-1/3/2.600(vc 1)        rmt   Up     Jan 24 11:00:26 2013           1
      Remote PE: 192.168.0.101, Negotiated control-word: Yes (Null)
      Incoming label: 299776, Outgoing label: 299776
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Local interface: ge-1/3/2.600, Status: Up, Encapsulation: VLAN
    Connection History:
        Jan 24 11:00:26 2013  status update timer
        Jan 24 11:00:26 2013  PE route changed
        Jan 24 11:00:26 2013  Out lbl Update                    299776
        Jan 24 11:00:26 2013  In lbl Update                     299776
        Jan 24 11:00:26 2013  loc intf up                 ge-1/3/2.600
Neighbor: 192.168.0.102
    Interface                 Type  St     Time last up          # Up trans
    ge-1/3/2.600(vc 2)        rmt   HS     -----                       ----
```

```
      Remote PE: 192.168.0.102, Negotiated control-word: Yes (Null)
      Incoming label: 299792, Outgoing label: 299776
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000020, Neighbor PW status code: 0x00000000
      Local interface: ge-1/3/2.600, Status: Up, Encapsulation: VLAN
user@PE1> show l2circuit connections extensive

Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  HS -- Hot-standby Connection
XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.168.0.100
    Interface                 Type  St     Time last up          # Up trans
    lt-1/2/0.600(vc 1)        rmt   Up     Jan 24 11:06:36 2013           1
      Remote PE: 192.168.0.100, Negotiated control-word: Yes (Null)
      Incoming label: 299776, Outgoing label: 299776
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Local interface: lt-1/2/0.600, Status: Up, Encapsulation: VLAN
    Connection History:
        Jan 24 11:06:36 2013  status update timer
        Jan 24 11:06:36 2013  PE route changed
        Jan 24 11:06:36 2013  Out lbl Update                    299776
        Jan 24 11:06:36 2013  In lbl Update                     299776
        Jan 24 11:06:36 2013  loc intf up                 lt-1/2/0.600
user@PE2> show l2circuit connections extensive

Layer-2 Circuit Connections:
```

```
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  HS -- Hot-standby Connection
XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.168.0.100
    Interface                 Type  St     Time last up          # Up trans
    lt-1/2/0.600(vc 2)        rmt   Up     Jan 24 10:55:31 2013           1
      Remote PE: 192.168.0.100, Negotiated control-word: Yes (Null)
      Incoming label: 299776, Outgoing label: 299792
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000020
      Local interface: lt-1/2/0.600, Status: Up, Encapsulation: VLAN
    Connection History:
        Jan 24 10:55:31 2013  status update timer
        Jan 24 10:55:31 2013  PE route changed
        Jan 24 10:55:31 2013  Out lbl Update                    299792
        Jan 24 10:55:31 2013  In lbl Update                     299776
        Jan 24 10:55:31 2013  loc intf up                 lt-1/2/0.600
```

Meaning

From the perspective of Device PE1 and Device PE2, a single Layer 2 circuit is established toward access routers, so there is no standby device information in the CLI output of the `show l2circuit connections` command. Note that no timing and flapping information is provided for the VC acting as the hot-standby. Junos OS only allows these counters to be tracked for the active VC.

Checking the Policy Conditions

Purpose

On the PE devices, verify the state of the different conditions defined as part of the Layer3 VPN's egress policy, where 10.41.0.0/24 corresponds to the logical tunnel (y) subnet.

Action

From operational mode, enter the `show policy conditions detail` command.

```
user@PE1> show policy conditions detail
Configured conditions:
Condition primary (static), event: Existence of a route in a specific routing table
Dependent routes:
 10.41.0.0/24, generation 8
 192.168.0.104/32, generation 8

Condition standby (static), event: Existence of a route in a specific routing table
Dependent routes:
None

Condition tables:
Table mpls.0, generation 0, dependencies 0, If-route-exists conditions: primary (static) standby
(static)
Table l3vpn.inet.0, generation 12, dependencies 2
```

```
user@PE2> show policy conditions detail
Configured conditions:
Condition primary (static), event: Existence of a route in a specific routing table
Dependent routes:
 10.41.0.0/24, generation 18

Condition standby (static), event: Existence of a route in a specific routing table
Dependent routes:
 10.41.0.0/24, generation 18

Condition tables:
Table mpls.0, generation 0, dependencies 0, If-route-exists conditions: primary (static) standby
```

```
(static)
Table l3vpn.inet.0, generation 367, dependencies 2
```

RELATED DOCUMENTATION

Understanding Pseudowire Redundancy Mobile Backhaul Scenarios | 382

Extension of Pseudowire Redundancy Condition Logic to Pseudowire Service Logical Interface Overview

IN THIS SECTION

- Sample Topology | 419
- Functionality | 420
- Policy Condition for Pseudowire Service Logical Interfaces | 420

The pseudowire redundancy feature for mobile backhaul scenarios uses logical tunnel (lt-) paired interfaces as the stitching between Layer 2 and Layer 3 domains. This feature now includes the MPLS pseudowire service logical interface to terminate subscriber interfaces using the ps0.0 interface as the stitching between Layer 2 and Layer 3 domains. This topic describes the functional details of the pseudowire redundancy feature using the ps0.0 interface, which extends the policy logic used in the logical tunnel interfaces.

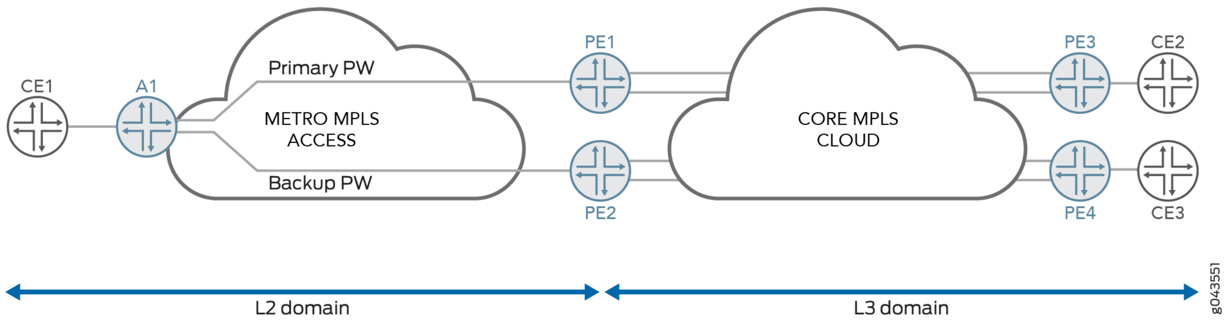
A pseudowire logical device and its related pseudowire logical interfaces are dependent on the state of the underlying logical transport interface device, which is either the Layer 2 VPN or Layer 2 circuit.

NOTE: We recommend that you use `unit 0` to represent the transport logical interface for the pseudowire device. Non-zero unit numbers represent *service* logical interfaces used for pseudowire subscriber interfaces.

Sample Topology

Figure 32 on page 420 on page 1 shows the stitching of Layer 2 and Layer 3 domains between the MPLS access node and the MPLS core. The primary or backup pseudowire on the MPLS access side is terminated at the provider edge devices (PE1 and PE2) at the pseudowire logical transport interface (ps0.0). The corresponding pseudowire logical service interfaces (ps0.1 to ps0.n) at the core MPLS cloud are connected to the Layer 3 domain, and these pseudowire logical service interfaces are configured in Layer 3 VPN routing instances.

Figure 32: Sample Topology for Pseudowire Redundancy



This topology results in a Layer 2 circuit across the MPLS access node and provider edge routers, with the pseudowire logical transport interface (ps0.0) acting as the local interface of the Layer 2 circuit terminating at the PE routers.

Functionality

Figure 32 on page 420 on page 1 shows the functional details of pseudowire redundancy with events between the devices. A1 is the MPLS access node that initiates the primary and backup Layer 2 circuits to the provider edge devices (PE1 and PE2). The Layer 2 circuit is terminated on provider edge devices and then stitched to the Layer 3 VPN.

The functional flow is as follows:

- Create the primary and backup Layer 2 circuit at access node A1.
- Detect both the primary and backup path, advertise the local preference, and stitch the Layer 2 circuit and Layer 3 VPN at the provider edge devices (PE1 and PE2).

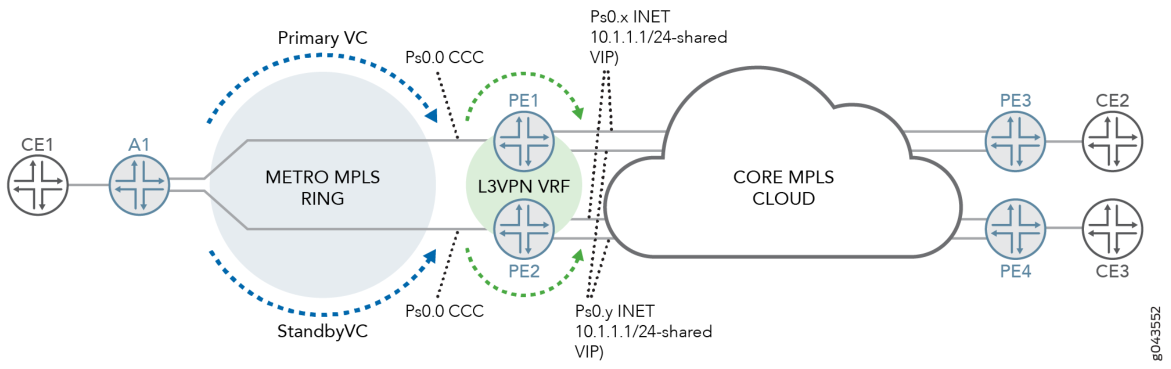
The following pseudowire code is used to notify the standby status from the access node to provider edge devices:

- L2CKT_PW_STATUS_PW_FWD_STDBY flag with 0x00000020.

Policy Condition for Pseudowire Service Logical Interfaces

The policy condition uses pseudowire service logical interfaces to stitch the Layer 2 and Layer 3 domains. Provider edge devices (PE1 and PE2) detect both primary and standby virtual circuits on the metro MPLS side. The primary virtual circuit is stitched to Layer 3 domain at the services side PE1 and PE2 towards the MPLS core. See Figure 33 on page 421 on page 2.

Figure 33: Pseudowire Service Logical Interface for Pseudowire Redundancy Solution



A primary Layer 2 circuit exists between access node A1 and provider edge device PE1, and a standby Layer 2 circuit exists between the same access node A1 and provider edge device PE2. The pseudowire service on the transport logical interface (ps0.0) is the local interface for the Layer 2 circuit at PE1 and PE2. At PE1 and PE2, there are multiple Layer 3 VPN instances; for example vrf-x and vrf-y.

The pseudowire service on service logical interfaces ps0.x and ps0.y are configured for the vrf-x and vrf-y routing instances respectively. For example, when the traffic with VLAN ID x originates from the access node to PE1 or PE2 on the Layer 2 circuit, it exits through the pseudowire service on the transport logical interface (ps0.0). Then the pseudowire service on service logical interface ps0.x is selected and sent through the vrf-x instance.

When the pseudowire state is active, the aggregation provider edge device (PE1 or PE2) advertises the subnet of the attachment circuit with the higher local preference value, which is indicated by the user through a manually configured policy.

NOTE: When remote provider edge devices receive two inet-vpn prefixes corresponding to the subnet of the attachment circuit, the highest local preference prefix received determines the primary datapath to be elected.

See the following output:

```
[edit policy-options]
    condition primary {
    if-route-exists {
        address-family {
            ccc {
                ps0.0;
                table mpls.0;
            }
        }
    }
}
policy-statement l3vpn_export {
    term primary {
        from condition primary;
        then {
            local-preference add 300;
            community set l3vpn;
            accept;
        }
    }
}
```

NOTE: In `policy-statement` *`name`* , under *condition primary*, there is no need to configure the *peer unit*, as it is valid only for the logical tunnel interface. For the pseudowire service logical interface, one-to-many mapping is used.

RELATED DOCUMENTATION

Understanding Pseudowire Redundancy Mobile Backhaul Scenarios | **382**

Example: Configuring Pseudowire Redundancy in a Mobile Backhaul Scenario | **388**

Pseudowire Subscriber Logical Interfaces Overview

Configuring a Pseudowire Subscriber Logical Interface

Configuring the Transport Logical Interface for a Pseudowire Subscriber Logical Interface

CHAPTER 24

Configuring Load Balancing for Layer 2 Circuits

CHAPTER 25

Configuring Protection Features for Layer 2 Circuits

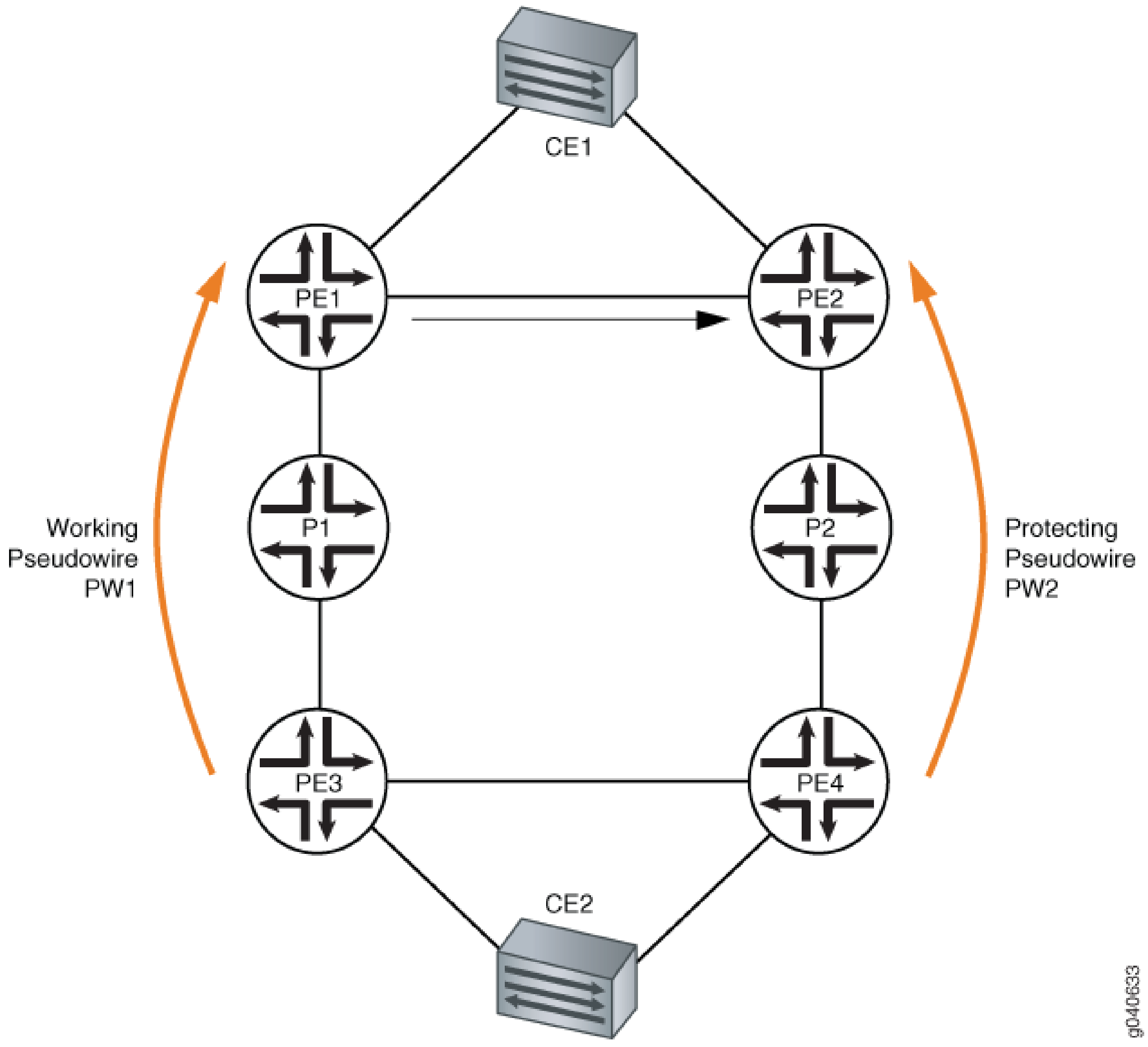
IN THIS CHAPTER

- Egress Protection LSPs for Layer 2 Circuits | **424**
- Configuring Egress Protection Service Mirroring for BGP Signaled Layer 2 Services | **426**
- Example: Configuring an Egress Protection LSP for a Layer 2 Circuit | **431**
- Example: Configuring Layer 2 Circuit Protect Interfaces | **447**
- Example: Configuring Layer 2 Circuit Switching Protection | **454**

Egress Protection LSPs for Layer 2 Circuits

An egress protection LSP provides link protection for link between PE routers and CE devices as illustrated in Figure 34 on page 425.

Figure 34: Egress Protection LSP



Device CE1 is multihomed to router PE1 and router PE2. Device CE2 is multihomed to router PE3 and router PE4. There are two paths connecting devices CE1 and CE2. The working path is CE2-PE3-P1-PE1-CE1, using pseudowire PW1. The protecting path is CE2-PE4-P2-PE2-CE1, using pseudowire PW2. Normally, traffic flows through the working path. When the end-to-end OAM between devices CE1 and CE2 detects a failure on the working path, traffic will be switched from the working path to the protecting path.

In the topology shown in Figure 34 on page 425, if there was a link or node failure in the core network (for example, a link failure from router P1 to PE1, from router PE3 to P1, or a node failure of router P1), MPLS fast reroute can be triggered on the transport LSPs between router PE3 and router PE1 to repair the connection within tens of milliseconds. Egress protection LSPs address the problem of when a link failure occurs at the edge of the network (for example, a link failure on router PE1 to device CE1).

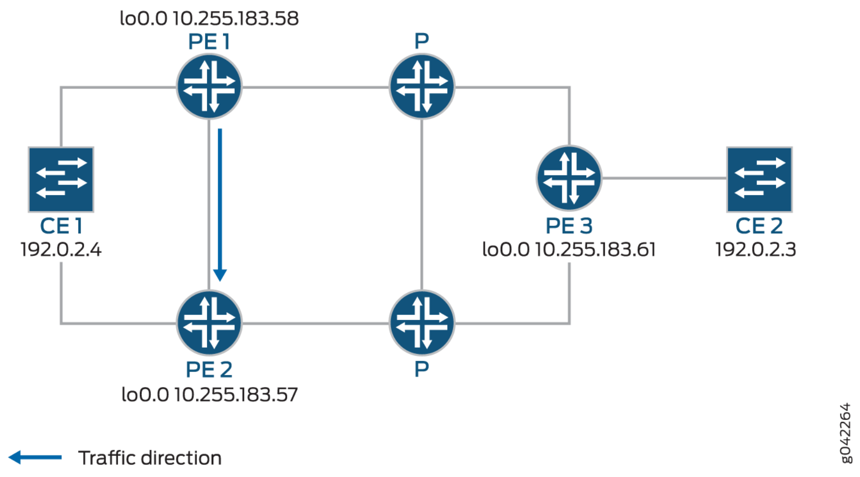
An egress protection LSP has been configured from router PE1 to router PE2. In the event of a link failure between router PE1 and device CE1, traffic can be switched to the egress protection LSP. Traffic from device CE2 can now be routed through path PE3-P1-PE1-PE2 to reach device CE1.

Configuring Egress Protection Service Mirroring for BGP Signaled Layer 2 Services

Starting in Junos OS Release 14.2, Junos OS supports the restoration of egress traffic when there is a link or node failure in the egress PE node. If there is a link or node failure in the core network, a protection mechanism such as MPLS fast reroute can be triggered on the transport LSPs between the PE routers to repair the connection within tens of milliseconds. An egress protection LSP addresses the problem of a node-link failure at the edge of the network (for example, a failure of a PE router).

Figure 1 shows a simplified topology of the use case that explains this feature.

Figure 35: Egress Protection LSP Configured from Router PE1 to Router PE2



CE1 is multihomed to PE1 and PE2. There are two paths connecting CE1 and CE2. The working path is CE2-PE3-P-PE1-CE1, via pseudowire PW21. The protecting path is CE2-PE3-P-PE2-CE1, via pseudowire PW22 Traffic is flowing through the working path under normal circumstances. When the end-to-end OAM between CE1 and CE2 detects failure on the working path, traffic will be switched from the working path to the protecting path. The end-to-end failure detection and recovery relies on control plane hence should be relatively slow. To achieve faster protection, local repair mechanisms similar to those used by MPLS fast reroute should be used. In Figure 1 above, if link or node failed in the core network (like link failure on P-PE1, P-PE3, or node failure on P), the MPLS fast reroute will happen on the transport LSPs between PE1 and PE3. The failure could be locally repaired within tens of milliseconds. However, if link or node failure happens at the edge (like link failure on PE3-CE2 or node failure on PE3), there is no local repair currently so we have to rely on the CE1-CE2 end-to-end protection to repair the failure.

- Device CE2—Traffic origin
- Router PE3—Ingress PE router
- Router PE1— (Primary) Egress PE router
- Router PE2—Protector PE router
- Device CE1—Traffic destination

When the link between CE1– PE1 goes downs, PE1 will briefly redirect that traffic towards CE1, to PE2. PE2 forwards it to CE1 until ingress router PE3 recalculates to forward the traffic to PE2.

Initially the traffic direction was; CE2 – PE3 – P – PE1 – CE1.

When the link between CE1– PE1 goes down, the traffic will be; CE2 – PE3 – P – PE1 – PE2 –CE1. PE3 then recalculates the path; CE2 – PE3 – P – PE2 – CE1.

1. Configure RSVP on PE1, PE2, and PE3.

```
[edit protocols]
user@PE1# set interface all
user@PE2# set interface all
user@PE3# set interface all
```

2. Configure MPLS.

```
[edit protocols mpls]
user@PE1# set interface all
user@PE2# set interface all
user@PE3# set interface all
```

3. Set PE1 as `primary` and PE2 as `protector` nodes.

```
[edit protocols mpls]
user@PE1# set egress-protection context-identifier address primary
user@PE2# set egress-protection context-identifier address protector
```

4. Enable `egress-protection` on PE1 and PE2.

```
[edit protocols bgp]
user@PE1# set group ibgp family l2vpn egress-protection
user@PE2# set group ibgp family l2vpn egress-protection
```

5. Configure LDP and ISIS on PE1, PE2, and PE3.

```
[edit protocols ldp]
user@PE1# set interface all
user@PE2# set interface all
user@PE3# set interface all
```

```
[edit protocols isis]
user@PE1# set interface all point-to-point
user@PE2# set interface all point-to-point
user@PE3# set interface all point-to-point
```

6. Configure a load balancing policy at PE1, PE2, and PE3.

```
[edit]
user@PE1# set policy-options policy-statement lb then load-balance per-packet
user@PE2# set policy-options policy-statement lb then load-balance per-packet
user@PE3# set policy-options policy-statement lb then load-balance per-packet
```

7. Configure the routing options at PE1, PE2, and PE3, to export routes based on the load balancing policy.

```
[edit]
user@PE1# set routing-options traceoptions file ro.log
user@PE1# set routing-options traceoptions flag normal
user@PE1# set routing-options traceoptions flag route
user@PE1# set routing-options autonomous-system 100
user@PE1# set routing-options forwarding-table export lb
```

```
[edit]
user@PE2# set routing-options traceoptions file ro.log
user@PE2# set routing-options traceoptions flag normal
```

```
user@PE2# set routing-options traceoptions flag route
user@PE2# set routing-options autonomous-system 100
user@PE2# set routing-options forwarding-table export lb
```

```
[edit]
user@PE3# set routing-options traceoptions file ro.log
user@PE3# set routing-options traceoptions flag normal
user@PE3# set routing-options traceoptions flag route
user@PE3# set routing-options autonomous-system 100
user@PE3# set routing-options forwarding-table export lb
```

8. Configure BGP at PE1 to advertise nrli from the routing instance with context-ID as next-hop.

```
[edit]
user@PE1# set routing-instances foo egress-protection context-identifier context-identifier
```

9. Configure l2vpn at PE1, PE2, and PE3

   At PE1:

```
[edit routing-instances]
foo {
    instance-type l2vpn;
    egress-protection {
        context-identifier {
            198.51.100.0;
        }
    }
    interface ge-2/0/2.0;
    route-distinguisher 10.255.183.58:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo {
                site-identifier 1;
                multi-homing;
                site-preference primary;
                interface ge-2/0/2.0 {
                    remote-site-id 2;
                }
            }
        }
    }
}
```

At PE2:

```
[edit routing-instances]
foo {
    instance-type l2vpn;
    egress-protection {
        protector;
    }
    interface ge-2/0/2.0;
    route-distinguisher 10.255.183.57:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo{
                site-identifier 1;
                multi-homing;
                site-preference backup;
                interface ge-2/0/2.0 {
                    remote-site-id 2;
                }
            }
        }
    }
}
```

At PE3:

```
[edit routing-instances]
foo {
    instance-type l2vpn;
    interface ge-2/1/2.0;
    route-distinguisher 10.255.183.61:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo {
                site-identifier 2;
                interface ge-2/1/2.0;
            }
        }
    }
}
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	Starting in Junos OS Release 14.2, Junos OS supports the restoration of egress traffic when there is a link or node failure in the egress PE node.

Example: Configuring an Egress Protection LSP for a Layer 2 Circuit

IN THIS SECTION

- Requirements | 431
- Egress Protection LSP Overview | 432
- Egress Protection LSP Configuration | 434

This example shows how to configure an egress protection LSP.

Requirements

Egress protection LSPs are supported on Juniper Networks MX Series routers only. This requirement applies to the PE routers facilitating the egress protection LSP.

Egress Protection LSP Overview

IN THIS SECTION

- Topology | 433

If there is a link or node failure in the core network, a protection mechanism such as MPLS fast reroute can be triggered on the transport LSPs between the PE routers to repair the connection within tens of milliseconds. An egress protection LSP addresses the problem of when a link failure occurs at the edge of the network (for example, a link failure between a PE router and a CE device). Egress protection LSPs do not address the problem of a node failure at the edge of the network (for example, a failure of a PE router). An egress protection LSP is an RSVP-signaled ultimate hop popping LSP.

This example includes the following configuration concepts and statements that are unique to the configuration of an egress protection LSP:

- `context-identifier`—Specifies an IPV4 address used to define the pair of PE routers participating in the egress protection LSP. The context identifier is used to assign an identifier to the protector PE router. The identifier is propagated to the other PE routers participating in the network, making it possible for the protected egress PE router to signal the egress protection LSP to the protector PE router.

- `egress-protection`—Configures the protector information for the protected Layer 2 circuit and configures the protector Layer 2 circuit at the `[edit protocols l2circuit]` hierarchy level. Configures an LSP as an egress protection LSP at the `[edit protocols mpls label-switched-path` *`lsp-name`*`]` hierarchy level. It also configures the context identifier at the `[edit protocols mpls]` hierarchy level.

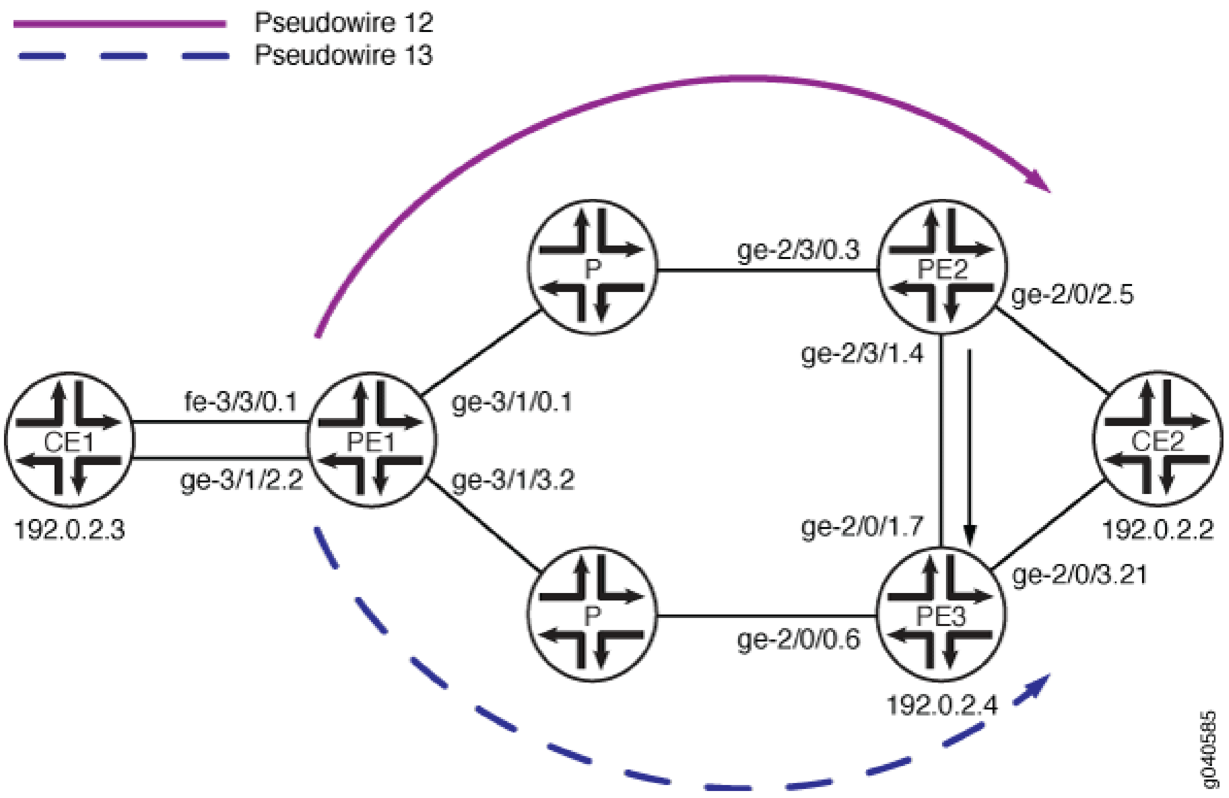
- `protected-l2circuit`—Specifies which Layer 2 circuit is to be protected by the egress protect LSP. This statement includes the following sub-statements: `ingress-pe`, `egress-pe`, and `virtual-circuit-id`. These sub-statements specify the address of the PE router at the ingress of the Layer 2 circuit, the address of the PE router at the egress of the Layer 2 circuit, and the Layer 2 circuit's identifier respectively.

- `protector-interface`—Specify the interface used by the egress protection LSP. In the event of a local link failure to a CE device, the egress protect LSP uses the interface specified to communicate with the protector PE router.

- `protector-pe`—Specify the IPv4 address of the protector PE router. The protector PE router must have a connection to the same CE device as the protected PE router for the egress protect LSP to function. This statement includes the following sub-statements: `context-identifier` and `lsp`. The `lsp` statement specifies the LSP to be used as the actual egress protection LSP.

Topology

Figure 36: Egress Protection LSP Configured from Router PE2 to Router PE3



Pseudowires are configured along two paths, one from router PE1 to router PE2 (pseudowire 12) and one from router PE1 to router PE3 (pseudowire 13). In the event of a failure on the link between router PE2 and device CE2, traffic is switched to the egress protection LSP configured between router PE2 and router PE3 (the protector PE router):

- Device CE1—Traffic origin
- Router PE1—Ingress PE router
- Router PE2—Egress PE router
- Router PE3—Protector PE router
- Device CE2—Traffic destintation

This example shows how to configure routers PE1, PE2, and PE3.

Egress Protection LSP Configuration

IN THIS SECTION

- CLI Quick Configuration | 434
- Step-by-Step Procedure | 436
- Results | 442

CLI Quick Configuration

To quickly configure an egress protection LSP, copy the following commands into a text file, modify the interface configurations to match your equipment, remove any line breaks, and then paste the commands into the CLI. This group of set commands is for router PE1.

```
set protocols rsvp interface ge-3/1/0.1
set protocols rsvp interface ge-3/1/3.2
set protocols mpls interface ge-3/1/0.1
set protocols mpls interface ge-3/1/3.2
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-3/1/0.1
set protocols ospf area 0.0.0.0 interface ge-3/1/3.2
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-3/1/0.1
set protocols ldp interface ge-3/1/3.2
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 192.0.2.3 interface fe-3/3/0.1 virtual-circuit-id 32
set protocols l2circuit neighbor 192.0.2.3 interface fe-3/3/0.1 egress-protection protector-
interface ge-3/1/2.2
set protocols l2circuit neighbor 192.0.2.4 interface ge-3/1/2.2 virtual-circuit-id 33
set policy-options policy-statement load-balance-example then load-balance per-packet
set routing-options router-id 192.0.2.2
set routing-options forwarding-table export load-balance-example
```

To quickly configure an egress protection LSP, copy the following commands into a text file, modify the interface configurations to match your equipment, remove any line breaks, and then paste the commands into the CLI. This group of set commands is for router PE2.

```
[edit]
set protocols rsvp tunnel-services
set protocols rsvp interface ge-2/3/0.3
set protocols rsvp interface ge-2/3/1.4 link-protection
set protocols ldp interface ge-2/3/0.3
set protocols ldp interface ge-2/3/1.4
set protocols ldp interface lo0.0
set protocols ldp upstream-label-assignment
set protocols mpls label-switched-path protected-lsp to 192.0.2.5
set protocols mpls label-switched-path protected-lsp egress-protection
set protocols mpls interface ge-2/3/0.3
set protocols mpls interface ge-2/3/1.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/3/0.3
set protocols ospf area 0.0.0.0 interface ge-2/3/1.4
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/2.5 virtual-circuit-id 23
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/2.5 egress-protection protector-pe
192.0.2.4
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/2.5 egress-protection protector-pe
context-identifier 192.0.2.5
set policy-options policy-statement load-balance-example then load-balance per-packet
set routing-options router-id 192.0.2.3
set routing-options forwarding-table export load-balance-example
```

To quickly configure an egress protection LSP, copy the following commands into a text file, modify the interface configurations to match your equipment, remove any line breaks, and then paste the commands into the CLI. This group of set commands is for router PE3.

```
set protocols rsvp tunnel-services
set protocols rsvp interface ge-2/0/0.6
set protocols rsvp interface ge-2/0/1.7
set protocols mpls interface ge-2/0/0.6
set protocols mpls interface ge-2/0/1.7
set protocols mpls egress-protection context-identifier 192.0.2.5 protector
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/0/0.6
set protocols ospf area 0.0.0.0 interface ge-2/0/1.7
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-2/0/0.6
set protocols ldp interface ge-2/0/1.7
set protocols ldp interface lo0.0
set protocols ldp upstream-label-assignment
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/3.21 virtual-circuit-id 42
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/3.21 egress-protection protected-
l2circuit PW1
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/3.21 egress-protection protected-
l2circuit ingress-pe 192.0.2.2
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/3.21 egress-protection protected-
l2circuit egress-pe 192.0.2.3
set protocols l2circuit neighbor 192.0.2.2 interface ge-2/0/3.21 egress-protection protected-
l2circuit virtual-circuit-id 31
```

Step-by-Step 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*.

To configure an egress protection LSP, complete the following steps for router PE1:

1. Configure RSVP. Include the interface linked to router PE2 and the interface linked to router PE3.

   ```
   [edit]
   user@PE1# edit protocols rsvp
   [edit protocols rsvp]
   user@PE1# set interface ge-3/1/0.1
   [edit protocols rsvp]
   user@PE1# set interface ge-3/1/3.2
   ```

2. Configure LDP. Include the interface linked to router PE2, the interface linked to router PE3, and the loopback interface.

   ```
   [edit]
   user@PE1# edit protocols ldp
   [edit protocols ldp]
   user@PE1# set interface ge-3/1/0.1
   ```

```
[edit protocols ldp]
user@PE1# set interface ge-3/1/3.2
[edit protocols ldp]
user@PE1# set interface lo0.0
```

3. Configure MPLS. Include the interface linked to router PE2 and the interface linked to router PE3.

```
[edit]
user@PE1# edit protocols mpls
[edit protocols mpls]
user@PE1# set interface ge-3/1/0.1
[edit protocols mpls]
user@PE1# set interface ge-3/1/3.2
```

4. Configure OSPF. Include the interface linked to router PE2, the interface linked to router PE3, and the loopback interface in the configuration for the OSPF area.

```
[edit]
user@PE1# edit protocols ospf
[edit protocols ospf]
user@PE1# set interface traffic-engineering
[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface ge-3/1/0.1
[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface ge-3/1/3.2
[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface lo0.0 passive
```

5. Configure Layer 2 circuits to use the egress protection LSP to protect against a link failure to device CE1.

```
[edit]
user@PE1# edit protocols l2circuit
[edit protocols l2circuit]
user@PE1# set neighbor 192.0.2.3 interface fe-3/3/0.1 virtual-circuit-id 32
[edit protocols l2circuit]
user@PE1# edit neighbor 192.0.2.3
[edit protocols l2circuit neighbor 192.0.2.3]
user@PE1# set interface fe-3/3/0.1 egress-protection protector-interface ge-3/1/2.2
```

```
[edit protocols l2circuit]
user@PE1# set neighbor 192.0.2.4 interface ge-3/1/2.2 virtual-circuit-id 33
```

6. Configure a load balancing policy.

```
[edit]
user@PE1# set policy-options policy-statement load-balance-example then load-balance per-
packet
```

7. Configure the routing options to export routes based on the load balancing policy.

```
[edit]
user@PE1# set routing-options router-id 192.0.2.2
[edit]
user@PE1# set routing-options forwarding-table export load-balance-example
```

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

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*.

To configure an egress protection LSP, complete the following steps for router PE2:

1. Configure RSVP. Include the interface linked to the ingress PE router and the interface linked to the CE device.

```
[edit]
user@PE2# edit protocols rsvp
[edit protocols rsvp]
user@PE2# set tunnel-services
[edit protocols rsvp]
user@PE2# set interface ge-2/3/0.3
[edit protocols rsvp]
user@PE2# set interface ge-2/3/1.4 link-protection
```

2. Configure LDP. Include the interface linked to the ingress PE router and the interface linked to the CE device.

```
[edit]
user@PE2# edit protocols ldp
[edit protocols ldp]
user@PE2# set interface ge-2/3/0.3
[edit protocols ldp]
user@PE2# set interface ge-2/3/1.4
[edit protocols ldp]
user@PE2# set interface lo0.0
[edit protocols ldp]
user@PE2# set upstream-label-assignment
```

3. Configure MPLS and the LSP which acts as the egress protection LSP.

```
[edit]
user@PE2# edit protocols mpls
[edit protocols mpls]
user@PE2# set interface ge-2/3/0.3
[edit protocols mpls]
user@PE2# set interface ge-2/3/1.4
[edit protocols mpls]
user@PE2# set label-switched-path protected-lsp to 192.0.2.5
[edit protocols mpls]
user@PE2# set label-switched-path protected-lsp egress-protection
```

4. Configure OSPF.

```
[edit]
user@PE2# edit protocols ospf
[edit protocols ospf]
user@PE2# set interface traffic-engineering
[edit protocols ospf]
user@PE2# set interface area 0.0.0.0 interface ge-2/3/0.3
[edit protocols ospf]
user@PE2# set interface area 0.0.0.0 interface ge-2/3/1.4
[edit protocols ospf]
user@PE2# set interface area 0.0.0.0 interface lo0.0 passive
```

5. Configure the Layer 2 circuit to use the egress protection LSP.

```
[edit]
user@PE2# edit protocols l2circuit
[edit protocols l2circuit]
user@PE2# set neighbor 192.0.2.2 interface ge-2/0/2.5 virtual-circuit-id 23
[edit protocols l2circuit]
user@PE2# edit neighbor 192.0.2.2
[edit protocols l2circuit neighbor 192.0.2.2]
user@PE2# set interface ge-2/0/2.5 egress-protectionprotector-pe 192.0.2.4
[edit protocols l2circuit neighbor 192.0.2.2]
user@PE2# set interface ge-2/0/2.5 egress-protection protector-pe context-identifier 192.0.2.5
```

6. Configure a load balancing policy.

```
[edit]
user@PE1# set policy-options policy-statement load-balance-example then load-balance per-
packet
```

7. Configure the routing options to export routes based on the load balancing policy.

```
[edit]
user@PE2# set routing-options router-id 192.0.2.3
[edit]
user@PE2# set routing-options forwarding-table export load-balance-example
```

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

Step-by-Step Procedure

To configure an egress protection LSP, complete the following steps for router PE3:

1. Configure RSVP. Include the interface linked to the ingress PE router and the interface linked to the CE device.

```
[edit]
user@PE3# edit protocols rsvp
[edit protocols rsvp]
user@PE3# set tunnel-services
```

```
[edit protocols rsvp]
user@PE3# set interface ge-2/0/0.6
[edit protocols rsvp]
user@PE3# set interface ge-2/0/1.7
```

2. Configure LDP. Include the interface linked to the ingress PE router and the interface linked to the CE device.

```
[edit]
user@PE3# edit protocols ldp
[edit protocols ldp]
user@PE3# set interface ge-2/0/0.6
[edit protocols ldp]
user@PE3# set interface ge-2/0/1.7
[edit protocols ldp]
user@PE3# set interface lo0.0
[edit protocols ldp]
user@PE3# set upstream-label-assignment
```

3. Configure MPLS and the LSP which acts as the egress protection LSP.

```
[edit]
user@PE3# edit protocols mpls
[edit protocols mpls]
user@PE3# set interface ge-2/0/0.6
[edit protocols mpls]
user@PE3# set interface ge-2/0/1.7
[edit protocols mpls]
user@PE3# set egress-protection context-identifier 192.0.2.5 protector
```

4. Configure OSPF.

```
[edit]
user@PE3# edit protocols ospf
[edit protocols ospf]
user@PE3# set interface traffic-engineering
[edit protocols ospf]
user@PE3# set area 0.0.0.0 interface ge-2/0/0.6
[edit protocols ospf]
user@PE3# set area 0.0.0.0 interface ge-2/0/1.7
```

```
[edit protocols ospf]
user@PE3# set area 0.0.0.0 interface lo0.0 passive
```

5. Configure the Layer 2 circuit to use the egress protection LSP.

```
[edit]
user@PE3# edit protocols l2circuit
[edit protocols l2circuit]
user@PE3# set neighbor 192.0.2.2 interface ge-2/0/3.21 virtual-circuit-id 42
[edit protocols l2circuit]
user@PE3# edit neighbor 192.0.2.2
[edit protocols l2circuit neighbor 192.0.2.2]
user@PE3# set interface ge-2/0/3.21 egress-protection protected-l2circuit ingress-pe 192.0.2.2
[edit protocols l2circuit neighbor 192.0.2.2]
user@PE3# set interface ge-2/0/3.21 egress-protection protected-l2circuit egress-pe 192.0.2.3
[edit protocols l2circuit neighbor 192.0.2.2]
user@PE3# set interface ge-2/0/3.21 egress-protection protected-l2circuitvirtual-circuit-id 31
```

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

Results

From configuration mode, confirm your configuration on router PE1 by entering the **show protocols**, **show policy-options**, and **show routing-options** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@PE1# show protocols
rsvp {
    interface ge-3/1/0.1;
    interface ge-3/1/3.2;
}
mpls {
    interface ge-3/1/0.1;
    interface ge-3/1/3.2;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-3/1/0.1;
        interface ge-3/1/3.2;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-3/1/0.1;
    interface ge-3/1/3.2;
    interface lo0.0;
}
l2circuit {
    neighbor 192.0.2.3 {
        interface fe-3/3/0.1 {
            virtual-circuit-id 32;
            egress-protection {
                protector-interface ge-3/1/2.2;
            }
        }
    }
    neighbor 192.0.2.4 {
        interface ge-3/1/2.2 {
            virtual-circuit-id 33;
        }
    }
}
[edit]
user@PE1# show policy-options
policy-statement load-balance-example {
    then {
        load-balance per-packet;
    }
}
[edit]
user@PE1# show routing-options
router-id 192.0.2.2;
forwarding-table {
    export load-balance-example;
}
```

From configuration mode, confirm your configuration on router PE2 by entering the **show protocols**, **show policy-options**, and **show routing-options** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@PE2# show protocols
rsvp {
    tunnel-services;
    interface ge-2/3/0.3;
    interface ge-2/3/1.4 {
        link-protection;
    }
}
mpls {
    label-switched-path protected-lsp {
        to 192.0.2.5;
        egress-protection;
    }
    interface ge-2/3/0.3;
    interface ge-2/3/1.4;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-2/3/0.3;
        interface ge-2/3/1.4;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-2/3/0.3;
    interface ge-2/3/1.4;
    interface lo0.0;
    upstream-label-assignment;
}
l2circuit {
    neighbor 192.0.2.2{
        interface ge-2/0/2.5 {
            virtual-circuit-id 23;
            egress-protection {
                protector-pe 192.0.2.4 context-identifier 192.0.2.5;
            }
        }
    }
}
```

```
[edit]
user@PE2# show policy-options
policy-options {
    policy-statement load-balance-example {
        then {
            load-balance per-packet;
        }
    }
}
```

```
[edit]
user@PE2# show routing-options
routing-options {
    router-id 192.0.2.3;
    forwarding-table {
        export load-balance-example;
    }
}
```

From configuration mode, confirm your configuration on router PE3 by entering the **show protocols** command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@PE3# show protocols
rsvp {
    tunnel-services;
    interface ge-2/0/0.6;
    interface ge-2/0/1.7;
}
mpls {
    interface ge-2/0/0.6;
    interface ge-2/0/1.7;
    egress-protection {
        context-identifier 192.0.2.5 {
            protector;
        }
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-2/0/0.6;
        interface ge-2/0/1.7;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-2/0/0.6;
    interface ge-2/0/1.7;
    interface lo0.0;
    upstream-label-assignment;
}
l2circuit {
    neighbor 192.0.2.2 {
        interface ge-2/0/3.21 {
            virtual-circuit-id 42;
            egress-protection {
                protected-l2circuit PW1 ingress-pe 192.0.2.2 egress-pe 192.0.2.3 virtual-circuit-
id 31;
            }
        }
    }
}
```

Example: Configuring Layer 2 Circuit Protect Interfaces

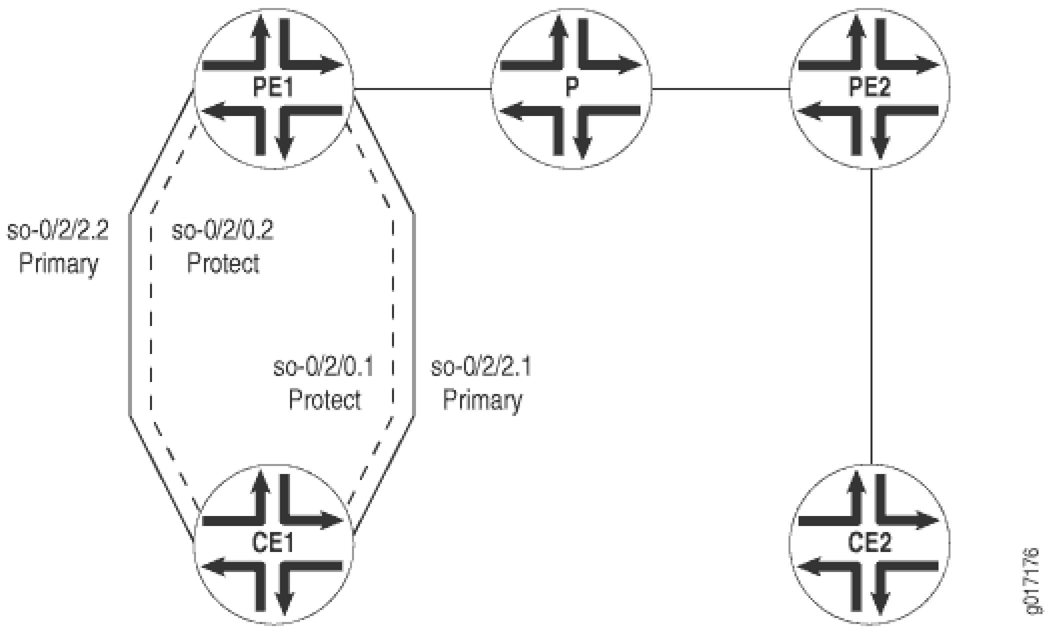
IN THIS SECTION

- Configuring Router PE1 | **448**
- Configuring Router PE2 | **450**
- Configuring Router CE1 | **452**
- Configuring Router CE2 | **453**

This example illustrates how you might configure a Layer 2 circuit with protect interfaces. Protect interfaces act as backups for their associated interfaces. The primary interface has priority over the protect interface and carries network traffic as long as it is functional. If the primary interface fails, the protect interface is activated. These interfaces can also share the same virtual path identifier (VPI) or virtual circuit identifier (VCI).

Figure 37 on page 447 shows the network topology used in this example.

Figure 37: Layer 2 Circuits Using Protect Interfaces



The following sections describe how to configure a Layer 2 circuit to use a protect interface:

Configuring Router PE1

Configure an interface for traffic to Router CE1 from Router PE1 at the [edit interfaces] hierarchy level:

```
[edit interfaces]
so-0/2/2 {
    description "Router CE1 so-0/2/2";
    no-keepalives;
    encapsulation frame-relay-ccc;
    unit 1 {
        encapsulation frame-relay-ccc;
        point-to-point;
        dlci 600;
    }
    unit 2 {
        encapsulation frame-relay-ccc;
        point-to-point;
        dlci 602;
    }
}
```

Configure an interface for traffic to Router CE1 from Router PE1 at the [edit interfaces] hierarchy level. Logical interface so-0/2/0.2 acts as the protect interface for so-0/2/2.2, and logical interface so-0/2/0.1 acts as the protect interface for so-0/2/2.1:

```
[edit interfaces]
so-0/2/0 {
    description "to Router CE1 so-0/3/0";
    no-keepalives;
    encapsulation frame-relay-ccc;
    unit 1 {
        encapsulation frame-relay-ccc;
        dlci 600;
    }
    unit 2 {
        encapsulation frame-relay-ccc;
        dlci 602;
    }
}
```

Configure an interface for traffic to Router PE2 from Router PE1 at the `[edit interfaces]` hierarchy level:

```
[edit interfaces]
so-0/2/1 {
    description "to Router PE2 so-1/0/1";
    unit 0 {
        family inet {
            address 192.0.2.0/24 {
                destination 192.0.2.4;
            }
        }
        family iso;
        family mpls;
    }
}
```

Configure an interface for traffic to Router PE2 from Router PE1 at the `[edit interfaces]` hierarchy level:

```
[edit interfaces]
so-0/2/3 {
    description "Router PE2 so-1/0/3";
    unit 0 {
        family inet;
        family iso;
        family mpls;
    }
    lo0 {
        unit 0 {
            family inet {
                address 198.51.100.0/24;
                address 10.100.40.200/32;
            }
            family iso {
                address 47.0005.80ff.f800.0000.0108.0001.1921.6800.4213.00;
            }
        }
    }
}
```

Configure the Layer 2 circuit by including the `l2circuit` statement at the `[edit protocols]` hierarchy level. The logical interfaces for the Layer 2 circuits and their corresponding protect interfaces are included here:

```
[edit protocols]
l2circuit {
    neighbor 10.100.40.210 {
        interface so-0/2/2.2 {
            protect-interface so-0/2/0.2;
            virtual-circuit-id 2;
            no-control-word;
        }
        interface so-0/2/2.1 {
            protect-interface so-0/2/0.1;
            virtual-circuit-id 1;
            no-control-word;
        }
    }
}
```

Configuring Router PE2

Configure an interface for traffic to Router CE2 from Router PE2:

```
[edit interfaces]
so-1/0/0 {
    description "to Router CE2 so-0/2/0";
    no-keepalives;
    encapsulation frame-relay-ccc;
    unit 1 {
        encapsulation frame-relay-ccc;
        point-to-point;
        dlci 700;
    }
    unit 2 {
        encapsulation frame-relay-ccc;
        point-to-point;
        dlci 702;
    }
}
```

Configure an interface for traffic to Router PE1 from Router PE2:

```
[edit interfaces]
so-1/0/1 {
    description "to Router PE1 so-0/2/1";
    unit 0 {
        family inet {
            address 192.0.2.4/32 {
                destination 192.0.2.22;
            }
        }
        family iso;
        family mpls;
    }
}
```

Configure an interface for traffic to Router PE1 from Router PE2:

```
[edit interfaces]
so-1/0/3 {
    description "to Router PE1 so-0/2/3";
    unit 0 {
        family inet;
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 198.51.100.0/24;
            address 10.100.40.210/32;
        }
        family iso {
            address 47.0005.80ff.f800.0000.0108.0001.1921.6800.4216.00;
        }
    }
}
```

Configure the Layer 2 circuit at the `[edit protocols]` hierarchy level:

```
[edit protocols]
l2circuit {
    neighbor 10.100.40.200 {
        interface so-1/0/0.1 {
            virtual-circuit-id 1;
            no-control-word;
        }
        interface so-1/0/0.2 {
            virtual-circuit-id 2;
            no-control-word;
        }
    }
}
```

Configuring Router CE1

Configure an interface for traffic to Router PE1 from Router CE1:

```
[edit interfaces]
so-0/3/0 {
    description "to Router PE1 so-0/2/0";
    no-keepalives;
    encapsulation frame-relay;
    unit 1 {
        dlci 601;
        family inet {
            address 203.0.113.1;
        }
    }
}
```

Configure an interface for traffic to Router PE1 from Router CE1:

```
[edit interfaces]
so-0/3/1 {
    description "Router PE1 so-0/2/2";
    no-keepalives;
    encapsulation frame-relay;
    unit 0 {
        dlci 600;
        family inet {
            address 10.10.10.1/24;
            address 203.0.113.2/24;
        }
        family iso;
        family mpls;
    }
    unit 2 {
        dlci 602;
        family inet {
            address 203.0.113.3/24;
        }
    }
}
```

Configuring Router CE2

Configure an interface for traffic to Router PE2 from Router CE2:

```
[edit interfaces]
so-0/2/0 {
    description "to Router PE2 so-1/0/0";
    no-keepalives;
    encapsulation frame-relay;
    unit 1 {
        dlci 700;
        family inet {
            address 10.10.10.2/24;
            address 203.0.113.5/24;
            address 203.0.113.6/24;
        }
    }
    unit 2 {
        dlci 702;
        family inet {
            address 203.0.113.7/24;
        }
    }
}
```

Example: Configuring Layer 2 Circuit Switching Protection

IN THIS SECTION

- Requirements | 454
- Overview | 455
- Configuration | 456

Unlike Layer 2 circuit protect interfaces (see "Example: Configuring Layer 2 Circuit Protect Interfaces" on page 447), which provide traffic protection for paths configured between the PE routers and CE routers, Layer 2 circuit switching protection provides traffic protection for the paths configured between the PE routers. In the event the path used by a Layer 2 circuit fails, traffic can be switched to an alternate path (or protection path). Switching protection is supported for locally switched Layer 2 circuits and provides 1 to 1 protection for each Layer 2 circuit interface.

When you enable Layer 2 circuit switching protection, each Layer 2 circuit interface requires the following paths:

- Working path—Used by the Layer 2 circuit when working normally.
- Protection path—Used by the Layer 2 circuit when the working path fails.

Requirements

This example uses the following hardware and software components:

- MX Series 5G Universal Routing Platforms
- Junos OS Release 12.3

Overview

IN THIS SECTION

- Topology | 455

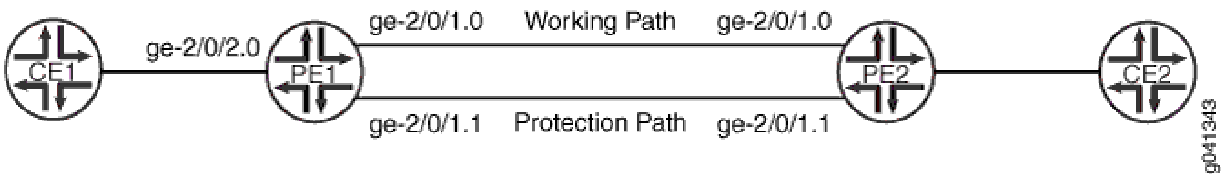
Each working path can be configured to have a either a protection path routed directly to the neighboring PE router (as shown in Figure 38 on page 455) or indirectly using a pseudowire configured through an intermediate PE router (as shown in Figure 39 on page 456 and Figure 40 on page 456). The protection path provides failure protection for the traffic flowing between the PE routers. Ethernet OAM monitors the status of these paths. When OAM detects a failure, it reroutes the traffic from the failed working path to the protection path. You can configure OAM to revert the traffic automatically to the working path when it is restored. You can also manually switch traffic between the working path, the protection path, and back.

NOTE: Non-stop routing (*NSR*) and graceful routing engine switchover (*GRES*) do not support Layer 2 circuit switching protection.

Topology

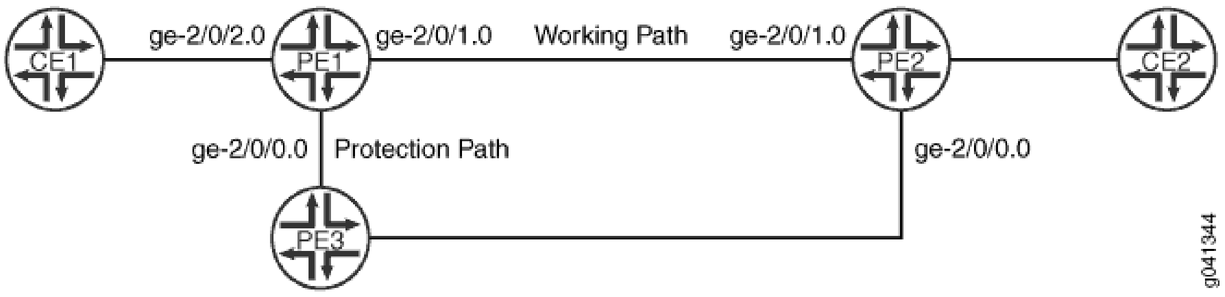
Figure 38 on page 455 illustrates Layer 2 circuit local switching. There are two OAM sessions running between Router PE1 and Router PE2. One OAM session is configured over the working path and the other is configured over the protection path.

Figure 38: Connection Protection Enabled Between Router PE1 and Router PE2



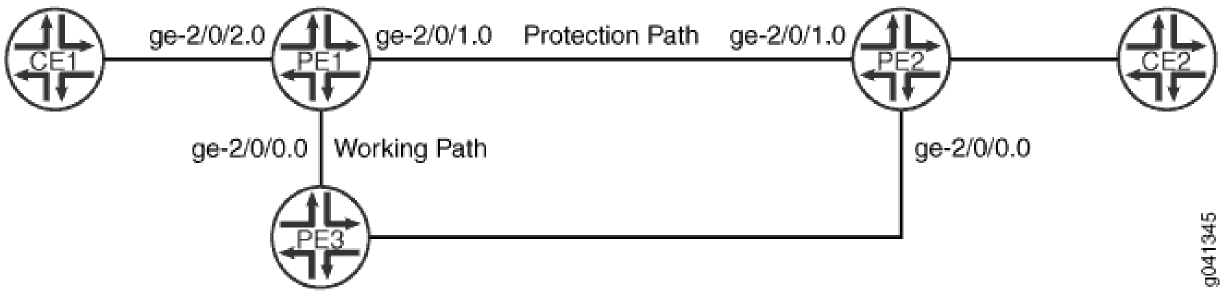
In Figure 39 on page 456 and Figure 40 on page 456, there are two OAM sessions running between Router PE1 and Router PE2. For Figure 2, one OAM session is configured over the working path between Router PE1 and Router PE2. The other OAM session is configured over the protection path between Router PE1 and Router PE3 to Router PE2.

Figure 39: Connection Protection Using a Pseudowire Configured through Router PE3 as the Protection Path



For Figure 40 on page 456, one OAM session is configured over the working path, the pseudowire between Router PE1 and Router PE3, then to Router PE2. The other OAM session is configured on the protect path between Router PE1 and Router PE2.

Figure 40: Connection Protection Using a Pseudowire Configured through Router PE3 as the Working Path



Configuration

IN THIS SECTION

- Configuring Connection Protection Between Two PE Routers | **457**
- Verifying that OAM CFM Connections are Active | **461**
- Configuring Connection Protection Using Another PE Router for the Protection Path | **462**
- Verifying that OAM CFM Connections are Active | **467**
- Configuring Connection Protection Using an Another PE Router for the Working Path | **468**
- Verifying that OAM CFM Connections are Active | **472**

The following sections describe how to configure each of the variations of Layer 2 circuit connection protection:

Configuring Connection Protection Between Two PE Routers

Step-by-Step Procedure

To configure Layer 2 Circuit switching protection as shown in Figure 38 on page 455 on Router PE1:

1. Configure the Layer 2 circuit on Router PE1.

```
[edit protocols l2circuit]
user@PE1# set local-switching interface ge-2/0/2.0 connection-protection
user@PE1# set local-switching interface ge-2/0/2.0 end-interface interface ge-2/0/1.0
user@PE1# set local-switching interface ge-2/0/2.0 end-interface backup-interface ge-2/0/1.1
```

2. Configure the routing policy on Router PE1.

```
[edit policy-options]
user@PE1# set policy-statement protection-policy then load-balance per-packet
```

3. Enable the routing policy on Router PE1.

```
[edit routing-options]
user@PE1# set forwarding-table export protection-policy
```

4. Configure OAM on Router PE1. OAM is used to monitor the working path between Router PE1 and Router PE2. In the event of a failure on the working path, traffic is switched automatically to the protection path. A connectivity fault management (CFM) session is configured on the working path and on the protection path. Begin by configuring the OAM maintenance domain.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

5. Configure OAM on Router PE1 for the working path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
```

```
maintenance-association working continuity-check interval 100ms
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 interface ge-2/0/1.0
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 interface working
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 remote-mep 103
```

6. Configure OAM on Router PE1 for the protection path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection continuity-check interval 100ms
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 interface ge-2/0/1.1
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 interface protect
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 remote-mep 104
```

7. Configure the OAM maintenance domain on Router PE2.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

8. Configure OAM on Router PE2 for the working path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working continuity-check interval 100ms
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 interface ge-2/0/1.0
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 interface working
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 direction down
```

```
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 remote-mep 1000
```

9. Configure OAM on Router PE2 for the protection path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection continuity-check interval 100ms
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 interface ge-2/0/1.1
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 interface protect
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 direction down
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 remote-mep 1001
```

Results

From configuration mode on Router PE1, confirm your configuration by entering the **show protocols l2circuit**, **show policy-options**, **show routing-options**, and **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host> show protocols l2circuit
local-switching {
    interface ge-2/0/2.0 {
        connection-protection;
        end-interface {
            interface ge-2/0/1.0;
            backup-interface ge-2/0/1.1;
        }
    }
}
```

```
user@host> show policy-options
policy-statement protection-policy {
    then {
        load-balance per-packet;
    }
}
```

```
user@host> **show routing-options**
forwarding-table {
    export protection-policy;
}
```

```
user@host> **show protocols oam ethernet**
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            continuity-check {
                interval 100ms;
            }
            mep 1000 {
                interface ge-2/0/1.0 working;
                direction down;
                remote-mep 103;
            }
        }
        maintenance-association protection {
            continuity-check {
                interval 100ms;
            }
            mep 1001 {
                interface ge-2/0/1.1 protect;
                direction down;
                remote-mep 104;
            }
        }
    }
}
```

From configuration mode on Router PE2, confirm your configuration by entering the **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            continuity-check {
                interval 100ms;
            }
            mep 103 {
                interface ge-2/0/1.0 working;
                direction down;
                remote-mep 1000;
            }
        }
        maintenance-association protection {
            continuity-check {
                interval 100ms;
            }
            mep 104 {
                interface ge-2/0/1.1 protect;
                direction down;
                remote-mep 1001;
            }
        }
    }
}
```

Verifying that OAM CFM Connections are Active

Purpose

Verify that the CFM connections are active on each of the PE routers.

Action

Execute the following command on each of the PE routers.

1. Verify that the CFM working connection on Router PE1 is active.

```
user@ PE1> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

2. Verify that the CFM protect connection on Router PE1 is active

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

3. Verify that the CFM working connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

4. Verify that the CFM protect connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

Configuring Connection Protection Using Another PE Router for the Protection Path

Step-by-Step Procedure

To configure Layer 2 Circuit switching protection as shown in Figure 39 on page 456 on Router PE1:

1. Configure the Layer 2 circuit on Router PE1.

```
[edit protocols l2circuit]
user@PE1# set local-switching interface ge-2/0/2.0 connection-protection
user@PE1# set local-switching interface ge-2/0/2.0 backup-neighbor 192.0.2.2 virtual-
circuit-id 2
user@PE1# set local-switching interface ge-2/0/2.0 backup-neighbor 192.0.2.2 community
```

```
example
user@PE1# set local-switching interface ge-2/0/2.0 end-interface interface ge-2/0/1.0
```

2. Configure the routing policy on Router PE1.

```
[edit policy-options]
user@PE1# set policy-statement load-balance then load-balance per-packet
user@PE1# set policy-statement protection-policy term protect from community example
user@PE1# set policy-statement protection-policy term protect then install-nexthop lsp-
regex lsp-protect-*
```

3. Configure the community.

```
[edit policy-options]
user@PE1# set community example members 65100:10
```

4. Configure the routing options on Router PE1.

```
[edit routing-options]
user@PE1# set forwarding-table export load-balance
```

5. Configure OAM on Router PE1 to setup the maintenance domain. OAM is used to monitor the working path between Router PE1 and Router PE2. In the event of a failure on the working path, traffic is switched automatically to the protection path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

6. Configure OAM on Router PE1 for the working path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 interface ge-2/0/1.0
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 remote-mep 103
```

7. Configure OAM on Router PE1 for the protection path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 interface ge-2/0/0.0
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 remote-mep 104
```

8. Configure OAM on Router PE2 to setup the maintenance domain.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

9. Configure OAM on Router PE2 for the working path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 interface ge-2/0/1.0
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 direction down
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 remote-mep 1000
```

10. Configure OAM on Router PE2 for the protection path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 interface ge-2/0/0.0
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 direction down
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 remote-mep 1001
```

Results

From configuration mode on Router PE1, confirm your configuration by entering the **show protocols l2circuit**, **show policy-options**, **show routing-options**, and **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host> show protocols l2circuit
local-switching {
    interface ge-2/0/2.0 {
        connection-protection;
        backup-neighbor 192.0.2.2 {
            virtual-circuit-id 2;
            community example;
        }
        end-interface {
            interface ge-2/0/1.0;
        }
    }
}
```

```
user@host> show policy-options
community example members 65100:10;
policy-statement load-balance {
    then {
        load-balance per-packet;
    }
}
policy-statement protection-policy {
    term protect {
        from community example;
        then {
            install-nexthop lsp-regex lsp-protect-*;
        }
    }
}
```

```
user@host> show routing-options
forwarding-table {
    export load-balance;
}
```

```
user@host> show protocols oam ethernet
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            mep 1000 {
                interface ge-2/0/1.0;
                direction down;
                remote-mep 103;
            }
        }
        maintenance-association protection {
            mep 1001 {
                interface ge-2/0/0.0;
                direction down;
                remote-mep 104;
            }
        }
    }
}
```

From configuration mode on Router PE2, confirm your configuration by entering the **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            mep 103 {
                interface ge-2/0/1.0;
                direction down;
                remote-mep 1000;
            }
        }
        maintenance-association protection {
            mep 104 {
                interface ge-2/0/0.0;
                direction down;
                remote-mep 1001;
            }
        }
    }
}
```

Verifying that OAM CFM Connections are Active

Purpose

Verify that the CFM connections are active on each of the PE routers.

Action

Execute the following command on each of the PE routers.

1. Verify that the CFM working connection on Router PE1 is active.

```
user@ PE1> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

2. Verify that the CFM protect connection on Router PE1 is active

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

3. Verify that the CFM working connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

4. Verify that the CFM protect connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

Configuring Connection Protection Using an Another PE Router for the Working Path

Step-by-Step Procedure

To configure Layer 2 Circuit switching protection as shown in Figure 40 on page 456 on Router PE1:

1. Configure the Layer 2 circuit on Router PE1.

```
[edit protocols l2circuit]
user@PE1# set neighbor 192.0.2.2 interface ge-2/0/2.0 virtual-circuit-id 2
user@PE1# set neighbor 192.0.2.2 interface ge-2/0/2.0 community example
user@PE1# set neighbor 192.0.2.2 interface ge-2/0/2.0 connection-protection
user@PE1# set neighbor 192.0.2.2 interface ge-2/0/2.0 backup-neighbor 192.0.2.3 virtual-
circuit-id 3
user@PE1# set neighbor 192.0.2.2 interface ge-2/0/2.0 backup-neighbor 192.0.2.3 standby
```

2. Configure the policies on Router PE1.

```
[edit policy-options]
user@PE1# set policy-statement load-balance then load-balance per-packet
user@PE1# set policy-statement protection-policy term protect from community example
user@PE1# set policy-statement protection-policy term protect then install-nexthop lsp-
regex lsp-primary
```

3. Configure the community.

```
[edit policy-options]
user@PE1# set community example members 65100:10
```

4. Configure the routing options on Router PE1.

```
[edit routing-options]
user@PE1# set forwarding-table export load-balance
```

5. Configure OAM on Router PE1 to setup the maintenance domain. OAM is used to monitor the working path between Router PE1 and Router PE2. In the event of a failure on the working path, traffic is switched automatically to the protection path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

6. Configure OAM on Router PE1 for the working path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 interface ge-2/0/0.0
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 1000 remote-mep 103
```

7. Configure OAM on Router PE1 for the protection path.

```
[edit protocols oam ethernet]
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 interface ge-2/0/1.0
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 direction down
user@PE1# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 1001 remote-mep 104
```

8. Configure OAM on Router PE2 to setup the maintenance domain.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md level 5
```

9. Configure OAM on Router PE2 for the working path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 interface ge-2/0/0.0
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 direction down
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association working mep 103 remote-mep 1000
```

10. Configure OAM on Router PE2 for the protection path.

```
[edit protocols oam ethernet]
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 interface ge-2/0/1.0
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 direction down
user@PE2# set connectivity-fault-management maintenance-domain l2circuit-example-md
maintenance-association protection mep 104 remote-mep 1001
```

Results

From configuration mode on Router PE1, confirm your configuration by entering the **show protocols l2circuit**, **show policy-options**, **show routing-options**, and **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host> show protocols l2circuit
neighbor 192.0.2.2 {
    interface ge-2/0/2.0 {
        virtual-circuit-id 2;
        community example;
        connection-protection;
        backup-neighbor 192.0.2.3 {
            virtual-circuit-id 3;
            standby;
        }
    }
}
```

```
user@host> **show policy-options**
community example members 65100:10;
policy-statement load-balance {
    then {
        load-balance per-packet;
    }
}
    policy-statement protection-policy {
        term protect {
            from community example;
            then {
                install-nexthop lsp-regex lsp-primary;
            }
        }
    }
```

```
user@host> show routing-options
forwarding-table {
    export load-balance;
}
```

```
user@host> show protocols oam ethernet
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            mep 1000 {
                interface ge-2/0/0.0;
                direction down;
                remote-mep 103;
            }
        }
        maintenance-association protection {
            mep 1001 {
                interface ge-2/0/1.0;
                direction down;
                remote-mep 104;
            }
        }
    }
}
```

From configuration mode on Router PE2, confirm your configuration by entering the **show protocols oam ethernet** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
connectivity-fault-management {
    maintenance-domain l2circuit-example-md {
        level 5;
        maintenance-association working {
            mep 103 {
                interface ge-2/0/0.0;
                direction down;
                remote-mep 1000;
            }
        }
        maintenance-association protection {
            mep 104 {
                interface ge-2/0/1.0;
                direction down;
                remote-mep 1001;
            }
        }
    }
}
```

Verifying that OAM CFM Connections are Active

Purpose

Verify that the CFM connections are active on each of the PE routers.

Action

Execute the following command on each of the PE routers.

1. Verify that the CFM working connection on Router PE1 is active.

```
user@ PE1> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

2. Verify that the CFM protect connection on Router PE1 is active

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

3. Verify that the CFM working connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association working
Interface status: Active, Link status: Up
```

4. Verify that the CFM protect connection on Router PE2 is active.

```
user@ PE2> show oam ethernet connectivity-fault-management mep-database maintenance-domain
l2circuit-example-md maintenance-association protection
Interface status: Active, Link status: Up
```

RELATED DOCUMENTATION

Example: Configuring Layer 2 Circuit Protect Interfaces | 447

CHAPTER 26

Monitoring Layer 2 Circuits with BFD

IN THIS CHAPTER

- Configuring BFD for VCCV for Layer 2 Circuits | **474**
- Example: Configuring BFD for VCCV for Layer 2 Circuits | **477**

Configuring BFD for VCCV for Layer 2 Circuits

Bidirectional Forwarding Detection (BFD) support for virtual circuit connection verification (VCCV) allows you to configure a control channel for a pseudowire, in addition to the corresponding operations and management functions to be used over that control channel. BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures. This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. You can also use a ping to detect pseudowire failures. However, the processing resources required for a ping are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based).

Before you begin:

- Configure the device interfaces.

To configure BFD for VCCV:

1. Specify the threshold for the adaptation of the BFD session detection time.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection]
user@host# set detection-time threshold milliseconds
```

For example, to set a detection time threshold of 40 milliseconds for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection]
user@host# set detection-time threshold 40
```

2. Configure the virtual circuit ID for the Layer 2 circuit protocol.

```
[edit protocols l2circuit neighbor IP-address interface interface-name]
user@host# set virtual-circuit-id virtual-circuit-id
```

For example, to set the virtual circuit ID as 1 for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection]
user@host# set virtual-circuit-id 1
```

3. Configure the minimum interval after which the local routing device transmits hello packets and then expects to receive a reply from a neighbor with which it has established a BFD session for the Layer 2 circuit.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection]
user@host# set minimum-interval milliseconds
```

For example, to set a minimum interval of 300 milliseconds for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection]
user@host# set minimum-interval 300
```

4. Configure the minimum interval after which the local routing device must receive a reply from a neighbor with which it has established a BFD session for the Layer 2 circuit protocol.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection]
user@host# set minimum-receive-interval milliseconds
```

For example, to set a minimum receive interval of 10 milliseconds for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection]
user@host# set minimum-receive-interval 10
```

5. Configure the number of hello packets not received by a neighbor that causes the originating interface to be declared down for the Layer 2 circuit protocol.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection]
user@host# set multiplier number
```

For example, to set the multiplier as 3 for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection]
user@host# set multiplier 3
```

6. Configure to disable adaptation.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection]
user@host# set no-adaptation
```

7. Configure the minimum interval at which the local routing device transmits hello packets to a neighbor with which it has established a BFD session.

```
[edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
detection transmit-interval]
user@host# set minimum-interval milliseconds
```

For example, to set a minimum transmit interval of 5 milliseconds for OAM BFD liveness detection:

```
[edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection
transmit-intreval]
user@host# set minimum-interval 5
```

8. Specify the threshold for the adaptation of the BFD session transmit interval.

   ```
   [edit protocols l2circuit neighbor IP-address interface interface-name oam bfd-liveness-
   detection transmit-interval]
   user@host# set threshold milliseconds
   ```

   For example, to set a transmit interval threshold of 30 milliseconds for OAM BFD liveness detection:

   ```
   [edit protocols l2circuit neighbor 192.0.2.1 interface ge-1/1/9.0 oam bfd-liveness-detection
   transmit-intreval]
   user@host# set threshold 30
   ```

RELATED DOCUMENTATION

Example: Configuring BFD for VCCV for Layer 2 Circuits | 477

Example: Configuring BFD for VCCV for Layer 2 Circuits

IN THIS SECTION

- Requirements | 477
- Overview | 478
- Configuration | 479
- Verification | 485

This example shows how to configure BFD for VCCV for Layer 2 circuits which enables faster detection of failure in the data path.

Requirements

This example uses the following hardware and software components:

- Two MX Series 5G Universal Routing Platforms
- Junos OS Release 12.1 or later running on all devices

Overview

IN THIS SECTION

- Topology | 478

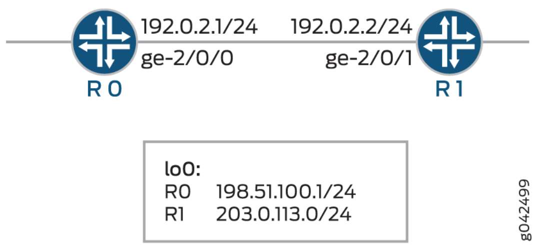
Starting with Junos OS Release 12.1, Bidirectional Forwarding Detection (BFD) support for virtual circuit connection verification (VCCV) allows you to configure a control channel for a pseudowire, in addition to the corresponding operations and management functions to be used over that control channel. BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures. This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. You can also use a ping to detect pseudowire failures. However, the processing resources required for a ping are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based).

To configure BFD for VCCV for Layer 2 circuits, configure the `oam` configuration statement at the `[edit protocols l2circuit neighbor` *`address`* `interface` *`interface-name`*`]` hierarchy level. The `control-channel` configuration statement at the `[edit routing-instances` *`routing-instance-name`* `protocols l2vpn oam]` hierarchy level does not apply to Layer 2 circuit configurations.

Topology

In the topology, BFD for VCCV for Layer 2 circuits is configured on Device R0.

Figure 41: BFD for VCCV for Layer 2 Circuits



Configuration

IN THIS SECTION

- CLI Quick Configuration | 479
- Configuring Device R0 | 481
- Results | 483

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 chassis redundancy graceful-switchover
set interfaces ge-1/1/9 vlan-tagging
set interfaces ge-1/1/9 encapsulation vlan-ccc
set interfaces ge-1/1/9 unit 0 encapsulation vlan-ccc
set interfaces ge-1/1/9 unit 0 vlan-id 512
set interfaces ge-2/0/0 unit 0 family inet address 192.0.2.1/24
set interfaces ge-2/0/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 198.51.100.0/24
set routing-options nonstop-routing
set routing-options static route 203.0.113.0/24 next-hop 192.0.2.2
set routing-options router-id 198.51.100.0
set protocols rsvp interface ge-2/0/0.0
set protocols mpls label-switched-path lsp1 to 203.0.113.0
set protocols mpls interface ge-2/0/0.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/0/0.0
set protocols ldp interface all
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 virtual-circuit-id 1
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
minimum-interval 300
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
minimum-receive-interval 10
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
```

```
multiplier 3
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
transmit-interval minimum-interval 5
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
transmit-interval threshold 30
set protocols l2circuit neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection
detection-time threshold 40
```

R1

```
set interfaces ge-1/1/9 vlan-tagging
set interfaces ge-1/1/9 encapsulation vlan-ccc
set interfaces ge-1/1/9 unit 0 encapsulation vlan-ccc
set interfaces ge-1/1/9 unit 0 vlan-id 512
set interfaces ge-2/0/1 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 203.0.113.0/24
set routing-options static route 198.51.100.0/24 next-hop 192.0.2.1
set routing-options router-id 203.0.113.0
set protocols rsvp interface ge-2/0/1.0
set protocols mpls label-switched-path lsp2 to 198.51.100.0
set protocols mpls interface ge-2/0/1.0
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/0/1.0
set protocols ldp interface all
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 virtual-circuit-id 1
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
minimum-interval 300
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
minimum-receive-interval 10
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
multiplier 3
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
transmit-interval minimum-interval 5
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
transmit-interval threshold 30
set protocols l2circuit neighbor 198.51.100.0 interface ge-1/1/9.0 oam bfd-liveness-detection
detection-time threshold 40
```

Configuring Device R0

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 Device R0:

NOTE: Repeat this procedure for Device R1 after modifying the appropriate interface names, addresses, and any other parameters for the device.

1. Configure graceful switchover redundancy.

```
[edit chassis]
user@R0# set redundancy graceful-switchover
```

2. Configure the interfaces.

```
[edit interfaces]
user@R0# set ge-1/1/9 vlan-tagging
user@R0# set ge-1/1/9 encapsulation vlan-ccc
user@R0# set ge-1/1/9 unit 0 encapsulation vlan-ccc
user@R0# set ge-1/1/9 unit 0 vlan-id 512
user@R0# set ge-2/0/0 unit 0 family inet address 192.0.2.1/24
user@R0# set ge-2/0/0 unit 0 family mpls
user@R0# set lo0 unit 0 family inet address 198.51.100.0/24
```

3. Configure the nonstop routing option, the static route, and the router ID routing options.

```
[edit routing-options]
user@R0# set nonstop-routing
user@R0# set static route 203.0.113.0/24 next-hop 192.0.2.2
user@R0# set router-id 198.51.100.0
```

4. Configure the RSVP protocol.

```
[edit protocols rsvp]
user@R0# set interface ge-2/0/0.0
```

5. Configure the MPLS protocol.

```
[edit protocols mpls]
user@R0# set label-switched-path lsp1 to 203.0.113.0
user@R0# set interface ge-2/0/0.0
```

6. Configure the OSPF protocol.

```
[edit protocols ospf]
user@R0# set traffic-engineering
user@R0# set area 0.0.0.0 interface ge-2/0/0.0
```

7. Configure the LDP protocol.

```
[edit protocols ldp]
user@R0# set interface all
```

8. Configure the virtual circuit ID for the neighbor of Layer 2 circuit protocols.

```
[edit protocols l2circuit]
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 virtual-circuit-id 1
```

9. Configure the oam attributes of the Layer 2 circuit protocol.

```
[edit protocols l2circuit]
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection minimum-
interval 300
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection minimum-
receive-interval 10
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection multiplier 3
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection transmit-
interval minimum-interval 5
```

```
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection transmit-
interval threshold 30
user@R0# set neighbor 203.0.113.0 interface ge-1/1/9.0 oam bfd-liveness-detection detection-
time threshold 40
```

Results

From configuration mode, confirm your configuration by entering the `show chassis`, `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@R0# show chassis
redundancy {
    graceful-switchover;
}
```

```
user@R0# show interfaces
ge-1/1/9 {
    vlan-tagging;
    encapsulation vlan-ccc;
    unit 0 {
        encapsulation vlan-ccc;
        vlan-id 512;
    }
}
ge-2/0/0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 198.51.100.0/24;
        }
    }
}
```

```
user@R0# **show protocols**
rsvp {
    interface ge-2/0/0.0;
}
mpls {
    label-switched-path lsp1 {
        to 203.0.113.0;
    }
    interface ge-2/0/0.0;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-2/0/0.0;
    }
}
ldp {
    interface all;
}
l2circuit {
    neighbor 203.0.113.0 {
        interface ge-1/1/9.0 {
            virtual-circuit-id 1;
            oam {
                bfd-liveness-detection {
                    minimum-interval 300;
                    minimum-receive-interval 10;
                    multiplier 3;
                    transmit-interval {
                        minimum-interval 5;
                        threshold 30;
                    }
                    detection-time {
                        threshold 40;
                    }
                }
            }
        }
    }
}
```

```
user@R0# show routing-options
nonstop-routing;
static {
    route 203.0.113.0/24 next-hop 192.0.2.2;
}
router-id 198.51.100.0;
```

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

Verification

IN THIS SECTION

- Verifying the Layer 2 Circuit Connections | **485**
- Verifying the BFD Session | **486**
- Verifying Detailed BFD Session Information | **487**

Verify that the configuration is working properly.

Verifying the Layer 2 Circuit Connections

Purpose

Verify the connections in a Layer 2 Circuit.

Action

From operational mode, run the `show l2circuit connections` command for Device R0.

```
user@R0> show l2circuit connections

Layer-2 Circuit Connections:

Legend for connection status (St)
```

```
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  HS -- Hot-standby Connection
XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 203.0.113.0
    Interface                 Type  St     Time last up          # Up trans
    ge-1/1/9.0(vc 1)          rmt   Up     Jun  2 03:19:44 2014           1
      Remote PE: 203.0.113.0, Negotiated control-word: Yes (Null)
      Incoming label: 299792, Outgoing label: 299792
      Negotiated PW status TLV: No
      Local interface: ge-1/1/9.0, Status: Up, Encapsulation: VLAN
      Flow Label Transmit: No, Flow Label Receive: No
      Flow Label Transmit: No, Flow Label Receive: No
```

Meaning

The output shows the Layer 2 virtual circuit information from Device R0 to its neighbor.

Verifying the BFD Session

Purpose

Verify the BFD session.

Action

From operational mode, run the `show bfd session` command for Device R0.

```
user@R0> show bfd session

                                                    Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier

203.0.113.7                Up        ge-2/0/0.0     0.030     0.010       3

1 sessions, 1 clients
Cumulative transmit rate 100.0 pps, cumulative receive rate 100.0 pps
```

Meaning

The output shows the address, and the interface on which the BFD session is active. The state *Up* indicates that the BFD session is up. The BFD session has a time interval of 30 milliseconds to detect BFD control packets , the transmitting system has a time interval of 10 milliseconds to send BFD control packets, and the transmitting system determines the detection time by multiplying *3* with the time interval. Total number of active BFD sessions and total number of clients that are hosting active BFD sessions. Cumulative transmit rate indicates the total number of BFD control packets transmitted, per second, on all active sessions and cumulative receive rate indicates the total number of BFD control packets received, per second, on all active sessions.

Verifying Detailed BFD Session Information

Purpose

Verify detailed BFD session information.

Action

From operational mode, run the `show bfd session extensive` command for Device R0.

```
user@R0> show bfd session extensive

                                                    Detect   Transmit
```

```
Address                  State     Interface      Time     Interval  Multiplier

203.0.113.7                Up        ge-2/0/0.0     0.030     0.010       3
 Client L2CKT-OAM, TX interval 0.005, RX interval 0.010
 Session up time 03:47:14
 Local diagnostic None, remote diagnostic None
 Remote state Up, version 1
 Replicated
 Session type: VCCV BFD
 Min async interval 0.005, min slow interval 1.000
 Adaptive async TX interval 0.005, RX interval 0.010
 Local min TX interval 0.005, minimum RX interval 0.010, multiplier 3
 Remote min TX interval 0.005, min RX interval 0.010, multiplier 3
 Threshold transmission interval 0.030, Threshold for detection time 0.040
 Local discriminator 20, remote discriminator 13004
 Echo mode disabled/inactive
 Remote is control-plane independent
 Neighbor address 203.0.113.0, Virtual circuit id 1
  Session ID: 0x0

1 sessions, 1 clients
Cumulative transmit rate 100.0 pps, cumulative receive rate 100.0 pps
```

Meaning

The output shows detailed information for the BFD session.

RELATED DOCUMENTATION

Configuring BFD for VCCV for Layer 2 Circuits | 474

CHAPTER 27

Troubleshooting Layer 2 Circuits

IN THIS CHAPTER

- Tracing Layer 2 Circuit Operations | 489

Tracing Layer 2 Circuit Operations

To trace the creation of and changes to Layer 2 circuits, include the `traceoptions` statement:

```
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
```

You can include this statement at the following hierarchy levels:

- `[edit protocols l2circuit]`
- `[edit logical-systems logical-system-name protocols l2circuit]`

Specify the following flags to trace the indicated operations on Layer 2 circuits:

- `connections`—Layer 2 circuit connections (events and state changes)
- `error`—Error conditions
- `FEC`—Layer 2 circuit advertisements received or sent using LDP
- `topology`—Layer 2 circuit topology changes caused by reconfiguration or advertisements received from other PE routers

6

PART

Configuring VPWS VPNs

Overview | 491

Configuring VPWS VPNs | 498

CHAPTER 28

Overview

IN THIS CHAPTER

- Understanding VPWS | **491**
- Supported VPWS Standards | **495**
- FAT Flow Labels Overview | **496**

Understanding VPWS

IN THIS SECTION

- Supported and Unsupported Features | **494**

Virtual private wire service (VPWS) Layer 2 VPNs employ Layer 2 services over MPLS to build a topology of point-to-point connections that connect end customer sites in a VPN. These Layer 2 VPNs provide an alternative to private networks that have been provisioned by means of dedicated leased lines or by means of Layer 2 virtual circuits. The service provisioned with these Layer 2 VPNs is known as VPWS. You configure a VPWS *instance* on each associated edge device for each VPWS Layer 2 VPN.

Traditional VPNs over Layer 2 circuits require the provisioning and maintenance of separate networks for IP and for VPN services. In contrast, *VPWS* enables the sharing of a provider's core network infrastructure between IP and Layer 2 VPN services, reducing the cost of providing those services.

Junos OS supports two types of VPWS Layer 2 VPNs:

- Kompella Layer 2 VPNs, which use BGP for autodiscovery and signaling.
- FEC 129 BGP autodiscovery for VPWS, which uses BGP for autodiscovery and LDP as the signaling protocol.

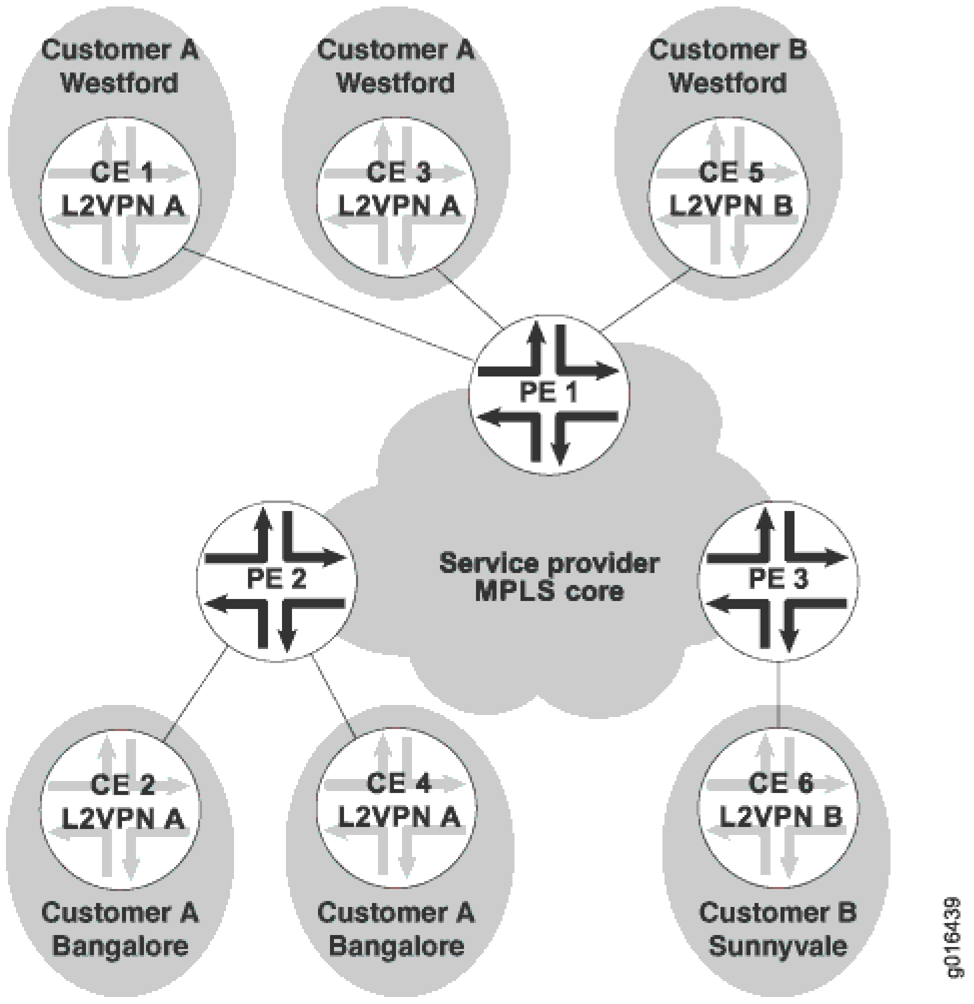
FEC 129 BGP autodiscovery for VPWS requires the `l2vpn-id`, `source-attachment-identifier`, and `target-attachment-identifier` statements. Kompella Layer 2 VPNs require the `site-identifier` and `remote-site-id` statements.

NOTE: VPWS creates pseudowires that emulate Layer 2 circuits. A virtual private LAN service (VPLS) network is similar to VPWS, but provides point-to-multipoint traffic forwarding in contrast to the VPWS Layer 2 VPN's point-to-point traffic forwarding. If you need point-to-multipoint service instead of point-to-point service, consider using VPLS instead of VPWS.

A VPWS Layer 2 VPN can have either a full-mesh or a hub-and-spoke topology. The tunneling mechanism in the core network typically is MPLS. However, VPWS can also use other tunneling protocols, such as GRE. VPWS is similar to Martini Layer 2 services over MPLS, and employs a similar encapsulation scheme for forwarding traffic.

Figure 42 on page 493 illustrates an example of a simple VPWS Layer 2 VPN topology.

Figure 42: VPWS Sample Topology



In this example, the service provider offers VPWS services to Customer A and Customer B. Customer A wants to create a full mesh of point-to-point links between Westford and Bengaluru. Customer B needs only a single point-to-point link between Westford and Sunnyvale. The service provider uses BGP and MPLS signaling in the core, and creates a set of unidirectional pseudowires at each provider edge (PE) device to separately cross-connect each customer's Layer 2 circuits.

In order to provision this service, the provider configures two VPWS Layer 2 VPNs, Layer 2 VPN A and Layer 2 VPN B. The circuit cross-connect (CCC) encapsulation type (`ethernet-ccc` or `vlan-ccc`) is configured for each VPWS Layer 2 VPN. All interfaces in a given VPWS Layer 2 VPN must be configured with the VPWS Layer 2 VPN's encapsulation type.

Local and remote site information for the interfaces identifies the cross-connect. Local cross-connects are supported when the interfaces that are connected belong to two different sites configured in the same VPWS instance and on the same PE device.

BGP advertises reachability for the VPNs. The BGP configuration is similar to that used for other VPN services, such as Layer 3 VPNs and VPLS. MPLS is configured to set up base LSPs to the remote PE devices similarly to the other VPN services.

Junos OS provides VPWS support the following configuration methods:

- Pseudowires are manually configured using Forwarding Equivalence Class (FEC) 128.
- Pseudowires are signaled by LDP using FEC 129. This arrangement reduces the configuration burden that is associated with statically configured Layer 2 circuits while still using LDP as the underlying signaling protocol.

Supported and Unsupported Features

Junos OS supports the following features with VPWS :

- Intra-AS VPWS functionality using BGP for autodiscovery and FEC 129 LDP for pseudowire signaling.
- *Graceful Routing Engine switchover*.
- Operation, administration, and maintenance (OAM) mechanisms, including Bidirectional Forwarding Detection and MPLS ping.
- FEC 128 LDP signaling with static configuration (in Junos OS this is configured within `protocols l2circuit`). With this option, there is no BGP autodiscovery.

Junos OS does not support the following VPWS functionality:

- Multihoming of customer sites to multiple PE devices using the BGP site model of multihoming.
- Terminating FEC 129 VPWS into a mesh group of an FEC 129 VPLS instance.
- Intra-AS VPWS functionality using BGP for autodiscovery and FEC 128 LDP for pseudowire signaling.
- FEC 129 VPWS without BGP autodiscovery.
- Static configuration of VPWS with FEC 129 signaling.
- *Nonstop active routing*.
- Multi-segment pseudowires.
- Interworking of FEC 128 and FEC 129 VPWS.
- Statically configured Layer 2 circuit-style pseudowire redundancy.
- Inter-AS deployments.

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | **913**

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | **938**

Supported VPWS Standards

Junos OS substantially supports the following RFCs, which define standards for VPWS and Layer 2 circuits.

- RFC 4447, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*

  Junos OS does not support Section 5.3, "The Generalized PWid FEC Element."

- RFC 4448, *Encapsulation Methods for Transport of Ethernet over MPLS Networks*
- RFC 6074, *Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*
- RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*
- RFC 6790, *The Use of Entropy Labels in MPLS Forwarding*

The following Internet drafts do not define standards, but provide information about Layer 2 technologies. The IETF classifies them as "Historic."

- Internet draft draft-martini-l2circuit-encap-mpls-11.txt, *Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks*

  Junos OS differs from the Internet draft in the following ways:

  - A packet with a sequence number of 0 (zero) is treated as out of sequence.
  - Any packet that does not have the next incremental sequence number is considered out of sequence.
  - When out-of-sequence packets arrive, the expected sequence number for the neighbor is set to the sequence number in the Layer 2 circuit control word.

- Internet draft draft-martini-l2circuit-trans-mpls-19.txt, *Transport of Layer 2 Frames Over MPLS*

RELATED DOCUMENTATION

Supported Carrier-of-Carriers and Interprovider VPN Standards

Supported Layer 2 VPN Standards | **146**

Supported Layer 3 VPN Standards
Supported Multicast VPN Standards
Supported VPLS Standards | **607**
Accessing Standards Documents on the Internet

FAT Flow Labels Overview

A pseudowire is a Layer 2 circuit or service that emulates the essential attributes of a telecommunications service over an MPLS packet-switched network (PSN). The pseudowire is intended to provide only the minimum necessary functionality to emulate the wire with the required resiliency requirements for the given service definition.

In an MPLS network, the flow-aware transport of pseudowires (FAT) flow label, as described in RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*, is used for load-balancing traffic across LDP-signaled pseudowires for virtual private LAN service (VPLS) and virtual private wire service (VPWS).

When the pseudowire is configured to use the FAT flow labels for load balancing, packets arriving at the ingress router are processed in the following sequence across the path of the pseudowire:

- The ingress router uses the contents of the inbound packet in the hash-key algorithm to calculate the flow-label value.
- The ingress router pushes the flow label to the label stack of the packet.
- The transit routers perform load balancing based only on the label stack.
- The egress router pops the flow label and forwards the packet to its destination.

For load balancing to work based on a flow-label configuration, a version of LDP that supports signaling extensions to use the flow label with pseudowires must be enabled on all routers. The LDP-signaling configuration is identical for VPLS and VPWS pseudowires.

FAT flow labels are supported on the following LDP-signaled forwarding-equivalence classes (FECs) for VPWS and VPLS pseudowires:

- FEC 128 for VPWS—LDP-signaled VPWS with neighbors that are statically configured (BGP autodiscovery is not supported).
- FEC 128 for VPLS—LDP-signaled VPLS with neighbors that are statically configured (BGP autodiscovery is not supported).
- FEC 129 for VPWS—LDP-signaled VPWS with BGP autodiscovery of neighbors.

- FEC 129 for VPLS—LDP-signaled VPLS with BGP autodiscovery of neighbors.

NOTE: Flow-aware transport of pseudowires (FAT) flow labels are not supported with Enhanced Dense Port Concentrators (DPCs), and flow labels should not be configured in a system with DPCs. Entropy labels are supported with DPCs with the following restrictions: entropy labels are stripped by DPCs, but entropy labels are not pushed by DPCs.

The interface parameter (Sub-TLV) is used both for FEC 128 and FEC 129 pseudowires. The sub-TLV defined for LDP contains the transmit (T) and receive (R) bits. The T bit advertises the ability to push the flow label. The R bit advertises the ability to pop the flow labels. By default, the signaling behavior of the provider edge (PE) router for any of these pseudowires is to advertise the T and R bits in the label set to 0.

The `flow-label-transmit` and `flow-label-receive` configuration statements provide the ability to set the T bit and R bit advertisement to 1 in the Sub-TLV field, which is part of the interface parameters of the FEC for the LDP label-mapping message. You can use these statements to control the pushing of the load-balancing label and the advertisement of the label to the routing peers in the control plane.

Alternatively, for FEC 128 VPWS pseudowires only, you can configure the following statements to statically configure flow label push and pop operations:

- `flow-label-receive-static` to statically pop the flow label on the pseudowire packets received from the remote PE router.
- `flow-label-transmit-static` to statically push the flow label on the pseudowire packets sent to the remote PE router.

RELATED DOCUMENTATION

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | 598

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | 770

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | 601

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | 772

CHAPTER 29

Configuring VPWS VPNs

IN THIS CHAPTER

- Understanding FEC 129 BGP Autodiscovery for VPWS | **498**
- Example: Configuring FEC 129 BGP Autodiscovery for VPWS | **501**
- Example: Configuring MPLS Egress Protection Service Mirroring for BGP Signaled Layer 2 Services | **519**
- Understanding Multisegment Pseudowire for FEC 129 | **543**
- Example: Configuring a Multisegment Pseudowire | **548**
- Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | **598**
- Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | **601**

Understanding FEC 129 BGP Autodiscovery for VPWS

IN THIS SECTION

- Supported Standards in FEC 129 BGP Autodiscovery for VPWS | **499**
- Routes and Routing Table Interaction in FEC 129 BGP Autodiscovery for VPWS | **499**
- Layer 2 VPN Behavior in FEC 129 BGP Autodiscovery for VPWS | **500**
- BGP Autodiscovery Behavior in FEC 129 BGP Autodiscovery for VPWS | **500**
- LDP Signaling Behavior in VPWS in FEC 129 BGP Autodiscovery for VPWS | **501**

The major functional components in a *VPWS* with FEC 129 are BGP, LDP, and the Layer 2 VPN module of Junos OS. BGP is responsible for distributing the local autodiscovery routes created on each PE device to all other PE devices. LDP is responsible for using the autodiscovery information provided by BGP to set up targeted LDP sessions over which to signal the pseudowires. The Layer 2 VPN is the glue that binds the BGP and LDP functionalities together.

Supported Standards in FEC 129 BGP Autodiscovery for VPWS

The relevant RFCs for this feature are as follows:

- RFC 4447, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*
- RFC 6074, *Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*

Routes and Routing Table Interaction in FEC 129 BGP Autodiscovery for VPWS

BGP, LDP, and Layer 2 VPNs interact through different types of routes installed in the *instance*.l2vpn.0 table. Theroutes that are present in the table are autodiscovery routes and pseudowire routes.

- Autodiscovery routes are used by BGP to allow autodiscovery of remote source access individual identifiers (SAIIs) (the sources of the point-to-point pseudowires) and PE device addresses. Autodiscovery routes are advertised when you configure the `l2vpn auto-discovery-only` address family.

  The format of the autodiscovery routes is a combination of the route distinguisher and the SAII. For example: 10.255.0.1:100:0.0.0.1/96 AD.

  Table 11 on page 499 lists the route elements and the number of associated bytes allocated to each element.

  Table 11: Autodiscovery Route Format

Route Element	Bytes
RD	8 bytes
SAII	4 bytes

  The `l2vpn-id` of the FEC 129 VPWS instance is attached to the route in a BGP extended community. One autodiscovery route is advertised for each source attachment identifier (SAI) in the instance.

- Pseudowire routes are installed by the Layer 2 VPN (local) and LDP (remote) to represent the bidirectional components of the pseudowire. For example: NoCtrlWord:5:100:200:2:0.0.0.1/176. The format of the routes is described in Table 12 on page 500.

Table 12: Pseudowire Route Format

Field Name	Field Description
Pseudowire type + control word bit	2 bytes
Remote PE address	4 bytes
Attachment group identifier (AGI) The AGI field of the pseudowire route is always set to the `l2vpn-id` of the instance.	8 bytes
SAII	4 bytes
Target attachment individual identifier (TAII)	4 bytes

Layer 2 VPN Behavior in FEC 129 BGP Autodiscovery for VPWS

A Layer 2 VPN installs a locally generated autodiscovery route into the instance.l2vpn.0 table for every SAII configured in an FEC 129 VPWS instance. The extended community containing the `l2vpn-id` is attached when the route is added to the instance.l2vpn.0 table.

For each autodiscovered SAII from a remote neighbor where the `l2vpn-id` matches the local `l2vpn-id` and the received SAII matches a locally configured TAII, the Layer 2 VPN obtains an MPLS label and generates a pseudowire route and adds it to the instance.l2vpn.0 table. The remote PE address is copied from the BGP protocol next hop for the autodiscovery route.

The Layer 2 VPN module of Junos OS is responsible for installing the forwarding routes into the mpls.0 table as usual.

BGP Autodiscovery Behavior in FEC 129 BGP Autodiscovery for VPWS

Local autodiscovery routes installed by the Layer 2 VPN in the instance.l2vpn.0 table are advertised by BGP to remote PE devices s`l2vpn auto-discovery-only` address family according to the instance and BGP export policies.

On the receiving side, BGP accepts autodiscovery routes from remote peers and installs them in the local bgp.l2vpn.0 table, if they are allowed by inbound policy. The route is installed, and a secondary route is imported into the instance.l2vpn.0 table when an import route target match between the route and instance is found.

LDP Signaling Behavior in VPWS in FEC 129 BGP Autodiscovery for VPWS

In the Junos OS implementation of LDP, the router monitors for routes from instance.l2vpn.0 for any instance configured for FEC 129 VPWS. These routes are identified by the `instance-type l2vpn` statement in the routing instance and the presence of the `l2vpn-id` statement.

When a BGP autodiscovery route is installed, LDP sets up a targeted session with the remote peer, where the peer address is identified as the protocol next hop of the BGP autodiscovery route.

When a pseudowire route is installed in the instance.l2vpn.0 table, LDP uses the parameters associated with the route to signal the creation of the pseudowire using FEC 129. Upon receiving an FEC 129 label mapping message from a remote peer, LDP installs the pseudowire route in the ldp.l2vpn.0 table.

Upon a successful `l2vpn-id` match with a configured FEC 129 VPWS instance, a secondary pseudowire route is imported to the instance.l2vpn.0 table. If an outgoing pseudowire has not already been set up when the incoming pseudowire signaling is received, LDP initiates the outgoing pseudowire creation as well.

RELATED DOCUMENTATION

Understanding VPWS | 491

Example: Configuring FEC 129 BGP Autodiscovery for VPWS | 501

Example: Configuring FEC 129 BGP Autodiscovery for VPWS

IN THIS SECTION

- Requirements | 502
- Overview | 502
- Configuration | 507
- Verification | 513

This example shows how to configure the virtual private wire service (VPWS), where remote provider edge (PE) devices are automatically discovered dynamically by BGP, and pseudowires are signaled by LDP using FEC 129. This arrangement reduces the configuration burden that is associated with statically configured Layer 2 circuits while still using LDP as the underlying signaling protocol.

Requirements

This example requires Junos OS Release 13.2 or later on the PE devices.

Overview

IN THIS SECTION

- Topology Diagram | **506**

Because VPWS is a point-to-point service, FEC 129 VPWS routing instances are configured as `instance-type l2vpn`. As with FEC 129 VPLS, FEC 129 VPWS uses the `l2vpn-id` statement to define the Layer 2 VPN of which the routing instance is a member. The presence of the `l2vpn-id` statement designates that FEC 129 LDP signaling is used for the routing instance. The absence of `l2vpn-id` indicates that BGP signaling is used instead.

The point-to-point nature of VPWS requires that you specify the source access individual identifier (SAII) and the target access individual identifier (TAII). This SAII-TAII pair defines a unique pseudowire between two PE devices.

The SAII is specified with the `source-attachment-identifier` statement within the FEC 129 VPWS routing instance. You configure the source attachment identifier and the interfaces to associate with that source attachment identifier. Under each interface, you can configure the TAII with the `target-attachment-identifier` statement. If the configured target identifier matches a source identifier advertised by a remote PE device by way of a BGP autodiscovery message, the pseudowire between that source-target pair is signaled. If there is no match between an advertised source identifier and the configured target identifier, the pseudowire is not established.

Sample: VPWS Configuration with Multiple Interfaces and Sites

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
        route-distinguisher 10.255.0.1:200;
        l2vpn-id l2vpn-id:100:200;
        vrf-target target:100:200;
        protocols l2vpn {
            site CUSTOMER-1 {
                source-attachment-identifier 1;
                interface ge-0/0/1.0 {
                    target-attachment-identifier 2;
                }
                interface ge-0/0/2.0 {
                    target-attachment-identifier 3;
                }
            }
        }
    }
}
```

You can configure multiple interfaces within a site, because each SAII-TAII pair defines a unique pseudowire, as shown with pseudowires 1-2 and 1-3 in the sample configuration. Both the source and target access identifiers are 4-byte numbers and can only be configured in FEC 129 VPWS instances where the `instance-type` is `l2vpn` and the `l2vpn-id` configuration statement is present.

You can specify the source and target identifiers as plain unsigned integers in the range 1 through 4,292,967,295.

The Layer 2 circuit and Layer 2 VPN services allow many optional parameters to be included on a per-pseudowire basis. FEC 129 VPWS allows such parameters as MTU settings, community tagging, and inclusion of a control word, as shown in this sample configuration:

Sample: VPWS Configuration with Optional Configuration Parameters

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
        route-distinguisher 10.255.0.1:200;
        l2vpn-id l2vpn-id:100:200;
        vrf-target target:100:200;
        protocols l2vpn {
            site CUSTOMER-1 {
                source-attachment-identifier 1;
                community COMM;
                control-word ;
                encapsulation-type ethernet;
                ignore-encapsulation-mismatch;
                ignore-mtu-mismatch;
                mtu 1500;
                no-control-word;
                interface ge-0/0/1.0 {
                    target-attachment-identifier 2;
                }
                interface ge-0/0/2.0 {
                    target-attachment-identifier 3;
                    community COMM;
                    control-word;
                    encapsulation-type ethernet;
                    ignore-encapsulation-mismatch;
                    ignore-mtu-mismatch;
                    mtu 1500;
                    no-control-word;
                }
            }
        }
    }
}
```

When configured within the site, the defined parameters affect any pseudowire originating from that site. When configured under an interface, the defined parameters affect that single specific pseudowire. This allows you to manipulate the parameters across all pseudowires associated with a particular local site in one place in the configuration.

Like other point-to-point services, the interfaces configured as members of the FEC 129 VPWS instance must be configured for CCC encapsulation and the CCC address family, as shown here:

```
interfaces {
    ge-0/0/1 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/2 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/3 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
}
```

You can use `vlan-ccc` instead of `ethernet-ccc`.

To support the basic FEC 129 VPWS functionality, the BGP sessions on the PE devices also need to be configured with the BGP `auto-discovery-only` address family to allow exchange of the autodiscovery routes. If traditional BGP VPLS or Layer 2 VPN service is also provisioned on the PE devices, the address family `l2vpn signaling` is also required, as shown here:

```
bgp {
    group pe {
        type internal;
        local-address 10.255.0.1;
        family l2vpn {
            auto-discovery-only;
            signaling;
        }
        neighbor 10.255.0.2;
        neighbor 10.255.0.3;
    }
}
```

The following configuration sample shows an FEC 129 VPWS routing instance with the operation, administration, and maintenance (OAM) (ping and BFD) configuration options:

Sample: VPWS Configuration with OAM

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        route-distinguisher 10.255.0.1:200;
```

```
            l2vpn-id l2vpn-id:100:200;
            vrf-target target:100:200;
            protocols l2vpn {
                oam {
                    ping-interval 600;
                    bfd-liveness-detection {
                        minimum-interval 200;
                    }
                }
                site CUSTOMER {
                    source-attachment-identifier 1;
                    oam {
                        ping-interval 600;
                        bfd-liveness-detection {
                            minimum-interval 200;
                        }
                    }
                    interface ge-0/0/1.0 {
                        oam {
                            ping-interval 600;
                            bfd-liveness-detection {
                                minimum-interval 200;
                            }
                        }
                        target-attachment-identifier 2;
                    }
                }
            }
        }
    }
```

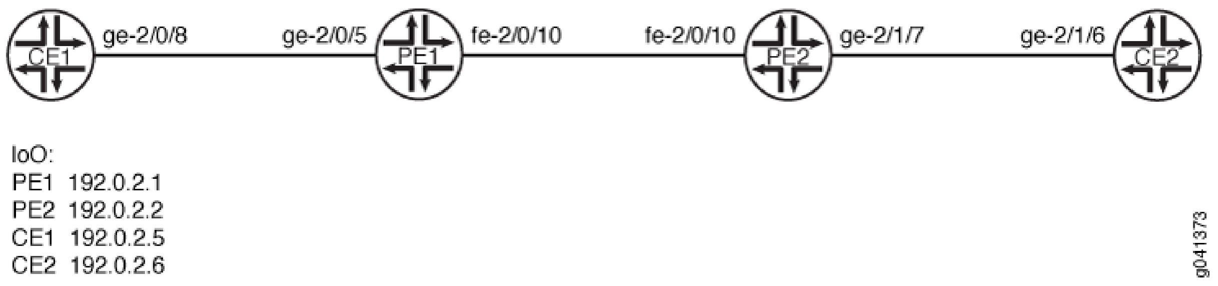
OAM options configured under `protocols l2vpn` apply to all sites and pseudowires in the routing instance. OAM options configured under a particular site apply to the pseudowires configured under that site. OAM options configured under a particular interface apply to the pseudowire configured under that interface.

Topology Diagram

Figure 43 on page 507 shows the topology used in this example.

This example uses a simple topology with two PE devices and two customer edge (CE) devices.

Figure 43: Simple VPWS Topology



"CLI Quick Configuration" on page 507 shows the configuration for all of the devices in Figure 43 on page 507. The section "Step-by-Step Procedure" on page 509 describes the steps on Device PE1 and Device PE2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 507
- Procedure | 509

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 CE1

```
set interfaces ge-2/0/8 unit 0 description CE1_to_PE1
set interfaces ge-2/0/8 unit 0 family inet address 172.16.0.1/24
set interfaces lo0 unit 0 family inet address 192.0.2.5/24
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 CE2

```
set interfaces ge-2/1/6 unit 0 description CE2_to_PE2
set interfaces ge-2/1/6 unit 0 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 192.0.2.6/24
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/6.0
```

Device PE1

```
set interfaces ge-2/0/5 encapsulation ethernet-ccc
set interfaces ge-2/0/5 unit 0 description PE1_to_CE1
set interfaces ge-2/0/5 unit 0 family ccc
set interfaces fe-2/0/10 unit 0 description to_PE2
set interfaces fe-2/0/10 unit 0 family inet address 10.0.0.1/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.1/24
set protocols mpls interface fe-2/0/10.0
set protocols bgp local-address 192.0.2.1
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances FEC129-VPWS instance-type l2vpn
set routing-instances FEC129-VPWS interface ge-2/0/5.0
set routing-instances FEC129-VPWS route-distinguisher 192.0.2.1:100
set routing-instances FEC129-VPWS l2vpn-id l2vpn-id:100:100
set routing-instances FEC129-VPWS vrf-target target:100:100
set routing-instances FEC129-VPWS protocols l2vpn site ONE source-attachment-identifier 1
set routing-instances FEC129-VPWS protocols l2vpn site ONE interface ge-2/0/5.0 target-
attachment-identifier 2
set routing-options autonomous-system 64510
```

Device PE2

```
set interfaces ge-2/1/7 encapsulation ethernet-ccc
set interfaces ge-2/1/7 unit 0 description PE2_to_CE2
set interfaces ge-2/1/7 unit 0 family ccc
set interfaces fe-2/0/10 unit 0 description to_PE1
set interfaces fe-2/0/10 unit 0 family inet address 10.0.0.2/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.2/24
set protocols mpls interface fe-2/0/10.0
set protocols bgp local-address 192.0.2.2
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.1
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances FEC129-VPWS instance-type l2vpn
set routing-instances FEC129-VPWS interface ge-2/1/7.0
set routing-instances FEC129-VPWS route-distinguisher 192.0.2.2:100
set routing-instances FEC129-VPWS l2vpn-id l2vpn-id:100:100
set routing-instances FEC129-VPWS vrf-target target:100:100
set routing-instances FEC129-VPWS protocols l2vpn site TWO source-attachment-identifier 2
set routing-instances FEC129-VPWS protocols l2vpn site TWO interface ge-2/1/7.0 target-
attachment-identifier 1
set routing-options autonomous-system 64510
```

Procedure

Step-by-Step Procedure

To configure a FEC 129 VPWS:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-2/0/5 encapsulation ethernet-ccc
user@PE1# set ge-2/0/5 unit 0 description PE1_to_CE1
```

```
user@PE1# set ge-2/0/5 unit 0 family ccc
user@PE1# set fe-2/0/10 unit 0 description to_PE2
user@PE1# set fe-2/0/10 unit 0 family inet address 10.0.0.1/30
user@PE1# set fe-2/0/10 unit 0 family mpls
user@PE1# set lo0 unit 0 family inet address 192.0.2.1/24
```

2. Configure MPLS on the core-facing interface.

```
[edit protocols mpls]
user@PE1# set interface fe-2/0/10.0
```

3. Configure BGP.

```
[edit protocols bgp]
user@PE1# set local-address 192.0.2.1
user@PE1# set group pe-pe type internal
user@PE1# set group pe-pe family l2vpn auto-discovery-only
user@PE1# set group pe-pe family l2vpn signaling
user@PE1# set group pe-pe neighbor 192.0.2.2
```

4. Configure an interior gateway protocol, such as IS-IS or OSPF.

   If you use OSPF, enable traffic engineering. Traffic engineering is supported by IS-IS by default.

```
[edit protocols ospf]
user@PE1# set traffic-engineering
user@PE1# set area 0.0.0.0 interface lo0.0 passive
user@PE1# set area 0.0.0.0 interface fe-2/0/10.0
```

5. Configure LDP on the core-facing interface and on the loopback interface.

```
[edit protocols ldp]
user@PE1# set interface fe-2/0/10.0
user@PE1# set interface lo0.0
```

6. Configure the VPWS routing instance.

LDP listens for routes from instance.l2vpn.0 for any instance configured for FEC 129 VPWS. These routes are identified by the `instance-type l2vpn` statement in the routing instance and the presence of the `l2vpn-id` statement.

Make sure that the `target-attachment-identifier` matches the `source-attachment-identifier` in the remote PE device's corresponding site. In this example, the pseudowire is established between Device PE1 and Device PE2. Device PE1 uses SAI 1 and TAI 2, while Device PE2 uses the opposite, SAI 2 and TAI 1.

```
[edit routing-instances FEC129-VPWS]
user@PE1# set instance-type l2vpn
user@PE1# set interface ge-2/0/5.0
user@PE1# set route-distinguisher 192.0.2.1:100
user@PE1# set l2vpn-id l2vpn-id:100:100
user@PE1# set vrf-target target:100:100
user@PE1# set protocols l2vpn site ONE source-attachment-identifier 1
user@PE1# set protocols l2vpn site ONE interface ge-2/0/5.0 target-attachment-identifier 2
```

7. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@PE1# set autonomous-system 64510
```

8. 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 interfaces`, `show protocols`, `show routing-instances`, and `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show interfaces
ge-2/0/5 {
    encapsulation ethernet-ccc;
    unit 0 {
        description PE1_to_CE1;
        family ccc;
    }
}
fe-2/0/10 {
    unit 1 {
        description to_PE2;
        family inet {
            address 10.0.0.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface fe-2/0/10.0;
}
bgp {
    local-address 192.0.2.1;
    group pe-pe {
        type internal;
        family l2vpn {
            auto-discovery-only;
            inactive: signaling;
        }
        neighbor 192.0.2.2;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-2/0/10.0;
    }
}
ldp {
    interface fe-2/0/10.0;
    interface lo0.0;
}
```

```
user@PE1# show routing-instances
FEC129-VPWS {
    instance-type l2vpn;
    interface ge-2/0/5.0;
    route-distinguisher 192.0.2.1:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        l2vpn {
            site ONE {
                source-attachment-identifier 1;
                interface ge-2/0/5.0 {
                    target-attachment-identifier 2;
                }
            }
        }
    }
}
```

```
user@PE1# show routing-options
autonomous-system 64510;
```

Verification

IN THIS SECTION

- Verifying the Routes | **514**
- Checking Connectivity Between the CE Devices | **516**
- Checking the VPWS Connections | **516**

● Checking Connectivity Between the PE Devices | 518

Confirm that the configuration is working properly.

Verifying the Routes

Purpose

Verify that the expected routes are learned.

Action

From operational mode, enter the `show route` command.

```
user@PE1> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.1/24         *[Direct/0] 6d 21:16:32
                    > via lo0.0
192.0.2.2/24         *[OSPF/10] 6d 21:15:31, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0
10.0.0.0/30        *[Direct/0] 6d 21:16:31
                    > via fe-2/0/10.0
10.0.0.1/32        *[Local/0] 6d 21:16:32
                      Local via fe-2/0/10.0
203.0.113.0/24       *[OSPF/10] 6d 21:16:34, metric 1
                      MultiRecv

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2/24         *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0

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

```
0                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
1                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
2                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
13                 *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
299808             *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0, Pop
299808(S=0)        *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0, Pop
299824             *[L2VPN/7] 5d 22:25:18
                    > via ge-2/0/5.0, Pop
ge-2/0/5.0         *[L2VPN/7] 5d 22:13:02, metric2 1
                    > to 10.0.0.2 via fe-2/0/10.0, Push 299872

bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2:100:0.0.0.2/96 AD
                   *[BGP/170] 6d 20:51:23, localpref 100, from 192.0.2.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-2/0/10.0

ldp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2:NoCtrlWord:5:100:100:0.0.0.2:0.0.0.1/176
                   *[LDP/9] 5d 22:13:02
                      Discard

FEC129-VPWS.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.1:100:0.0.0.1/96 AD
                   *[L2VPN/170] 6d 20:53:26, metric2 1
                      Indirect
192.0.2.2:100:0.0.0.2/96 AD
                   *[BGP/170] 6d 20:51:23, localpref 100, from 192.0.2.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-2/0/10.0
192.0.2.2:NoCtrlWord:5:100:100:0.0.0.1:0.0.0.2/176
```

```
                   *[L2VPN/7] 6d 20:51:23, metric2 1
                    > to 10.0.0.2 via fe-2/0/10.0
192.0.2.2:NoCtrlWord:5:100:100:0.0.0.2:0.0.0.1/176
                   *[LDP/9] 5d 22:13:02
                      Discard
```

Meaning

The output shows all the learned routes, including the autodiscovery (AD) routes.

Checking Connectivity Between the CE Devices

Purpose

Verify that Device CE1 can ping Device CE2.

Action

```
user@CE1> ping 192.0.2.6
PING 192.0.2.6 (192.0.2.6): 56 data bytes
64 bytes from 192.0.2.6: icmp_seq=0 ttl=64 time=0.679 ms
64 bytes from 192.0.2.6: icmp_seq=1 ttl=64 time=0.524 ms
^C
--- 192.0.2.6 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.524/0.602/0.679/0.078 ms
```

Meaning

The output shows that the VPWS is operational.

Checking the VPWS Connections

Purpose

Make sure that all of the FEC 129 VPWS connections come up correctly.

Action

```
user@PE1> show l2vpn connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: FEC129-VPWS
  L2vpn-id: 100:100
  Local source-attachment-id: 1 (ONE)
    Target-attachment-id      Type  St     Time last up          # Up trans
    2                         rmt   Up     Nov 28 16:16:14 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 299792, Outgoing label: 299792
      Local interface: ge-2/0/5.0, Status: Up, Encapsulation: ETHERNET
```

Meaning

As expected, the connection is up. The output includes the source attachment ID and the target attachment ID.

Checking Connectivity Between the PE Devices

Purpose

Verify that Device PE1 can ping Device PE2. The `ping mpls l2vpn fec129` command accepts SAIs and TAIs as integers or IP addresses and also allows you to use the CE-facing interface instead of the other parameters (`instance`, `local-id`, `remote-id`, `remote-pe-address`).

Action

```
user@PE1> ping mpls l2vpn fec129 instance FEC129-VPWS remote-id 2 remote-pe-address 192.0.2.2
local-id 1
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

```
user@PE1> ping mpls l2vpn fec129 interface ge-2/0/5.0
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Meaning

The output shows that the VPWS is operational.

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | 913

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938

Example: Configuring MPLS Egress Protection Service Mirroring for BGP Signaled Layer 2 Services

IN THIS SECTION

- Requirements | 519
- Overview | 519
- Configuration | 521
- Verification | 538

Starting in Junos OS Release 14.2, Junos OS supports the restoration of egress traffic when there is a link or node failure in the egress PE node. If there is a link or node failure in the core network, a protection mechanism such as MPLS fast reroute can be triggered on the transport LSPs between the PE routers to repair the connection within tens of milliseconds. An egress protection LSP addresses the problem of a node-link failure at the edge of the network (for example, a failure of a PE router).

This example shows how to configure link protection for BGP signaled Layer 2 services.

Requirements

MX Series Routers running Junos OS Release 14.2 or later.

Overview

IN THIS SECTION

- Topology | 520

If there is a link or node failure in the core network, a protection mechanism such as MPLS fast reroute can be triggered on the transport LSPs between the PE routers to repair the connection within tens of milliseconds. An egress protection LSP addresses the problem of a node-link failure at the edge of the network (for example, a failure of a PE router).

This example includes the following configuration concepts and statements that are unique to the configuration of an egress protection LSP:

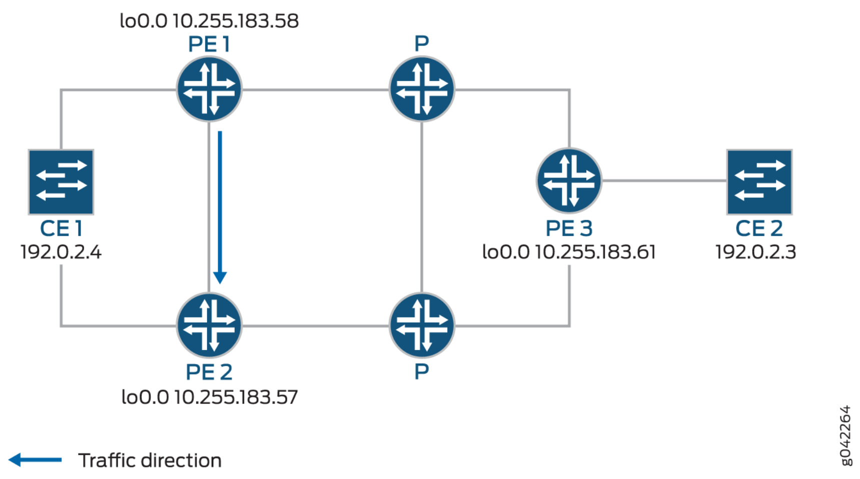
- `context-identifier`—Specifies an IPv4 or IPv6 address used to define the pair of PE routers participating in the egress protection LSP. It is assigned to each ordered pair of primary PE and the protector to facilitate protection establishment. This address is globally unique, or unique in the address space of the network where the primary PE and the protector reside.

- `egress-protection`—Configures the protector information for the protected Layer 2 circuit and configures the protector Layer 2 circuit at the `[edit protocols mpls]` hierarchy level. Configures an LSP as an egress protection LSP at the `[edit protocols mpls]` hierarchy level.

- `protector`—Configures the creation of standby pseudowires on the backup PE for link or node protection for the instance.

Topology

Figure 44: Egress Protection LSP Configured from Router PE1 to Router PE2



In the event of a failure of the egress PE Router PE1, traffic is switched to the egress protection LSP configured between Router PE1 and Router PE2 (the protector PE router):

- Device CE2—Traffic origin
- Router PE3—Ingress PE router
- Router PE1— (Primary) Egress PE router
- Router PE2—Protector PE router
- Device CE1—Traffic destination

When the link between CE1– PE1 goes downs, PE1 will briefly redirect that traffic toward CE1, to PE2. PE2 forwards it to CE1 until ingress router PE3 recalculates to forward the traffic to PE2.

Initially the traffic direction was: CE2 – PE3 – P – PE1 – CE1.

When the link between CE1– PE1 goes down, the traffic will be: CE2 – PE3 – P – PE1 – PE2 –CE1. PE3 then recalculates the path: CE2 – PE3 – P – PE2 – CE1.

This example shows how to configure routers PE1, PE2, and PE3.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 521
- Step-by-Step Procedure | 524
- Results | 531

CLI Quick Configuration

To quickly configure an egress protection LSP, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configurations, copy and then paste the commands into the CLI and enter `commit` from configuration mode.

PE1

```
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls egress-protection context-identifier 198.51.100.3 primary
set protocols mpls egress-protection context-identifier 198.51.100.3 advertise-mode stub-alias
set protocols mpls egress-protection traceoptions file ep size 100m
set protocols mpls egress-protection traceoptions flag all
set protocols bgp traceoptions file bgp.log world-readable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.183.58
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family l2vpn signaling egress-protection
set protocols bgp group ibgp neighbor 192.0.2.3
set protocols bgp group ibgp neighbor 192.0.2.4
set protocols isis traceoptions file isis-edge size 10m world-readable
set protocols isis traceoptions flag error
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface all point-to-point
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set policy-options policy-statement lb then load-balance per-packet
set routing-options traceoptions file ro.log
set routing-options traceoptions flag all
set routing-options traceoptions flag route
set routing-options autonomous-system 100
set routing-options forwarding-table export lb
set routing-instances foo instance-type l2vpn
set routing-instances foo egress-protection context-identifier 198.51.100.3
set routing-instances foo interface ge-2/0/2.0
set routing-instances foo route-distinguisher 10.255.183.58:1
set routing-instances foo vrf-target target:9000:1
set routing-instances foo protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances foo protocols l2vpn site foo site-identifier 1
set routing-instances foo protocols l2vpn site foo site-preference primary
set routing-instances foo protocols l2vpn site foo interface ge-2/0/2.0 remote-site-id 2
```

PE2

```
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls egress-protection context-identifier 198.51.100.3 protector
set protocols mpls egress-protection context-identifier 198.51.100.3 advertise-mode stub-alias
set protocols mpls egress-protection traceoptions file ep size 100m
set protocols mpls egress-protection traceoptions flag all
set protocols bgp traceoptions file bgp.log world-readable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.183.57
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family l2vpn signaling egress-protection
set protocols bgp group ibgp neighbor 192.0.2.3
set protocols bgp group ibgp neighbor 192.0.2.4
set protocols isis traceoptions file isis-edge size 10m world-readable
set protocols isis traceoptions flag error
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface all point-to-point
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set policy-options policy-statement lb then load-balance per-packet
set routing-options traceoptions file ro.log
set routing-options traceoptions flag normal
set routing-options traceoptions flag route
set routing-options autonomous-system 100
set routing-options forwarding-table export lb
set routing-instances foo instance-type l2vpn
set routing-instances foo egress-protection protector
set routing-instances foo interface ge-2/0/2.0
set routing-instances foo route-distinguisher 10.255.183.57:1
set routing-instances foo vrf-target target:9000:1
set routing-instances foo protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances foo protocols l2vpn site foo hot-standby
set routing-instances foo protocols l2vpn site foo site-identifier 1
set routing-instances foo protocols l2vpn site foo site-preference backup
set routing-instances foo protocols l2vpn site foo interface ge-2/0/2.0 remote-site-id 2
```

PE3

```
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp traceoptions file bgp.log world-readable
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.183.61
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 192.0.2.3
set protocols bgp group ibgp neighbor 192.0.2.4
set protocols isis traceoptions file isis-edge size 10m world-readable
set protocols isis traceoptions flag error
set protocols isis level 1 disable
set protocols isis level 2 wide-metrics-only
set protocols isis interface all point-to-point
set protocols isis interface all level 2 metric 10
set protocols isis interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set policy-options policy-statement lb then load-balance per-packet
set routing-options traceoptions file ro.log
set routing-options traceoptions flag normal
set routing-options traceoptions flag route
set routing-options autonomous-system 100
set routing-options forwarding-table export lb
set routing-instances foo instance-type l2vpn
set routing-instances foo interface ge-2/1/2.0
set routing-instances foo route-distinguisher 10.255.183.61:1
set routing-instances foo vrf-target target:9000:1
set routing-instances foo protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances foo protocols l2vpn site foo site-identifier 2
set routing-instances foo protocols l2vpn site foo interface ge-2/1/2.0 remote-site-id 1
```

Step-by-Step 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*.

To configure an egress protection LSP for router PE1:

1. Configure RSVP.

```
[edit protocols rsvp]
user@PE1# set interface all
user@PE1# set interface fxp0.0 disable
```

2. Configure MPLS to use the egress protection LSP to protect against a link failure to Device CE1.

```
[edit protocols mpls]
user@PE1# set interface all
user@PE1# set interface fxp0.0 disable
```

```
user@PE1# set egress-protection context-identifier 198.51.100.3 primary
user@PE1# set egress-protection context-identifier 198.51.100.3 advertise-mode stub-alias
user@PE1# set egress-protection traceoptions file ep size 100m
user@PE1# set egress-protection traceoptions flag all
```

3. Configure BGP.

```
[edit protocols bgp]
user@PE1# set traceoptions file bgp.log world-readable
user@PE1# set group ibgp type internal
user@PE1# set group ibgp local-address 10.255.183.58
user@PE1# set group ibgp family inet unicast
user@PE1# set group ibgp family l2vpn signaling egress-protection
user@PE1# set group ibgp neighbor 192.0.2.3
user@PE1# set group ibgp neighbor 192.0.2.4
```

4. Configure IS-IS.

```
[edit protocols isis]
user@PE1# set traceoptions file isis-edge size 10m world-readable
user@PE1# set traceoptions flag error
user@PE1# set level 1 disable
user@PE1# set level 2 wide-metrics-only
user@PE1# set interface all point-to-point
user@PE1# set interface all level 2 metric 10
user@PE1# set interface fxp0.0 disable
```

5. Configure LDP.

```
[edit protocols ldp]
user@PE1# set interface all
user@PE1# set interface fxp0.0 disable
```

6. Configure a load-balancing policy.

```
[edit]
user@PE1# set policy-options policy-statement lb then load-balance per-packet
```

7. Configure the routing options to export routes based on the load-balancing policy.

```
[edit routing-options]
user@PE1# set traceoptions file ro.log
user@PE1# set traceoptions flag all
user@PE1# set autonomous-system 100
user@PE1# set forwarding-table export lb
```

8. Configure BGP to advertise nrli from the routing instance with context-ID as next-hop.

```
[edit routing-instances]
user@PE1# set foo instance-type l2vpn
user@PE1# set foo egress-protection context-identifier 198.51.100.3
user@PE1# set foo interface ge-2/0/2.0
user@PE1# set foo route-distinguisher 10.255.183.58:1
user@PE1# set foo vrf-target target:9000:1
```

9. Configure l2vpn instance to use the egress LSP configured.

```
[edit routing-instances]
user@PE1# set foo protocols l2vpn encapsulation-type ethernet-vlan
user@PE1# set foo protocols l2vpn site foo site-identifier 1
user@PE1# set foo protocols l2vpn site foo site-preference primary
user@PE1# set foo protocols l2vpn site foo interface ge-2/0/2.0 remote-site-id 2
```

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

Step-by-Step Procedure

To configure an egress protection LSP for Router PE2:

1. Configure RSVP.

```
[edit protocols rsvp]
user@PE2# set interface all
user@PE2# set interface fxp0.0 disable
```

2. Configure MPLS and the LSP that acts as the egress protection LSP.

```
[edit protocols mpls]
user@PE2# set interface all
user@PE2# set interface fxp0.0 disable
user@PE2# set egress-protection context-identifier 198.51.100.3 protector
user@PE2# set egress-protection context-identifier 198.51.100.3 advertise-mode stub-alias
user@PE2# set egress-protection traceoptions file ep size 100m
user@PE2# set egress-protection traceoptions flag all
```

3. Configure BGP.

```
[edit protocols bgp]
user@PE2# set traceoptions file bgp.log world-readable
user@PE2# set group ibgp type internal
user@PE2# set group ibgp local-address 10.255.183.57
user@PE2# set group ibgp family inet unicast
user@PE2# set group ibgp family l2vpn signaling
user@PE2# set group ibgp family l2vpn egress-protection
user@PE2# set group ibgp neighbor 192.0.2.3
user@PE2# set group ibgp neighbor 192.0.2.4
```

4. Configure IS-IS.

```
[edit protocols isis]
user@PE2# set traceoptions file isis-edge size 10m world-readable
user@PE2# set traceoptions flag error
user@PE2# set level 1 disable
user@PE2# set level 2 wide-metrics-only
user@PE2# set interface all point-to-point
user@PE2# set interface all level 2 metric 10
user@PE2# set interface fxp0.0 disable
```

5. Configure LDP.

```
[edit protocols ldp]
user@PE2# set interface all
user@PE2# set interface fxp0.0 disable
```

6. Configure a load-balancing policy.

```
[edit]
user@PE2# set policy-options policy-statement lb then load-balance per-packet
```

7. Configure the routing options to export routes based on the load-balancing policy.

```
[edit routing-options]
user@PE2# set traceoptions file ro.log
user@PE2# set traceoptions flag all
user@PE2# set autonomous-system 100
user@PE2# set forwarding-table export lb
```

8. Configure BGP to advertise nrli from the routing instance with context-ID as next-hop.

```
[edit routing-instances]
user@PE2# set foo instance-type l2vpn
user@PE2# set foo egress-protection protector
user@PE2# set foo interface ge-2/0/2.0
user@PE2# set foo route-distinguisher 10.255.183.57:1
user@PE2# set foo vrf-target target:9000:1
```

9. Configure l2vpn instance to use the egress LSP configured.

```
[edit routing-instances]
user@PE2# set foo protocols l2vpn encapsulation-type ethernet-vlan
user@PE2# set foo protocols l2vpn site foo hot-standby
user@PE2# set foo protocols l2vpn site foo site-identifier 1
user@PE2# set foo protocols l2vpn site foo site-preference backup
user@PE2# set foo protocols l2vpn site foo interface ge-2/0/2.0 remote-site-id 2
```

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

Step-by-Step Procedure

To configure an egress protection LSP for Router PE3:

1. Configure RSVP.

```
[edit protocols rsvp]
user@PE3# set interface all
user@PE3# set interface fxp0.0 disable
```

2. Configure MPLS.

```
[edit protocols mpls]
user@PE3# set interface all
user@PE3# set interface fxp0.0 disable
```

3. Configure BGP.

```
[edit protocols bgp]
user@PE3# set traceoptions file bgp.log world-readable
user@PE3# set group ibgp type internal
user@PE3# set group ibgp local-address 10.255.183.61
user@PE3# set group ibgp family inet unicast
user@PE3# set group ibgp family l2vpn signaling
user@PE3# set group ibgp neighbor 192.0.2.3
user@PE3# set group ibgp neighbor 192.0.2.4
```

4. Configure IS-IS.

```
[edit protocols isis]
user@PE3# set traceoptions file isis-edge size 10m world-readable
user@PE3# set traceoptions flag error
user@PE3# set level 1 disable
user@PE3# set level 2 wide-metrics-only
user@PE3# set protocols isis interface all point-to-point
[edit protocols isis]
user@PE3# set protocols isis interface all level 2 metric 10
[edit protocols isis]
user@PE3# set protocols isis interface fxp0.0 disable
```

5. Configure LDP.

```
[edit protocols ldp]
user@PE3# set interface all
user@PE3# set interface fxp0.0 disable
```

6. Configure a load-balancing policy.

```
[edit]
user@PE3# set policy-options policy-statement lb then load-balance per-packet
```

7. Configure the routing options to export routes based on the load-balancing policy.

```
[edit routing-options]
user@PE3# set traceoptions file ro.log
user@PE3# set traceoptions flag normal
user@PE3# set traceoptions flag route
user@PE3# set autonomous-system 100
user@PE3# set forwarding-table export lb
```

8. Configure BGP to advertise nlri from the routing instance with context-ID as next-hop.

```
[edit]
user@PE3# set routing-instances foo instance-type l2vpn
user@PE3# set routing-instances foo interface ge-2/1/2.0
user@PE3# set routing-instances foo route-distinguisher 10.255.183.61:1
user@PE3# set routing-instances foo vrf-target target:9000:1
```

9. Configure l2vpn to specify the interface that connects to the site and the remote interface to which you want the specified interface to connect.

```
[edit routing-instances]
user@PE3# set foo protocols l2vpn encapsulation-type ethernet-vlan
user@PE3# set foo protocols l2vpn site foo site-identifier 2
user@PE3# set foo protocols l2vpn site foo interface ge-2/1/2.0 remote-site-id 1
```

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

Results

From configuration mode, confirm your configuration on Router PE1 by entering the **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.

```
[edit]
user@PE1# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
    egress-protection {
        context-identifier 198.51.100.3 {
            primary;
            advertise-mode stub-alias;
        }
        traceoptions {
            file ep size 100m;
            flag all;
        }
    }
}
bgp {
    traceoptions {
        file bgp.log world-readable;
    }
    group ibgp {
        type internal;
        local-address 10.255.183.58;
        family inet {
            unicast;
        }
        family l2vpn {
            signaling {
                egress-protection;
            }
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.4;
    }
}
isis {
    traceoptions {
        file isis-edge size 10m world-readable;
        flag error;
    }
    level 1 disable;
    level 2 wide-metrics-only;
    interface all {
        point-to-point;
        level 2 metric 10;
    }
    interface fxp0.0 {
        disable;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}

[edit]
user@PE1# show policy-options
policy-statement lb {
    then {
        load-balance per-packet;
    }
}
[edit]
user@PE1# show routing-options
traceoptions {
    file ro.log;
    flag all;
}
autonomous-system 100;
```

```
forwarding-table {
    export lb;
}

[edit]
user@PE1# show routing-instances
foo {
    instance-type l2vpn;
    egress-protection {
        context-identifier {
            198.51.100.3;
        }
    }
    interface ge-2/0/2.0;
    route-distinguisher 10.255.183.58:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo {
                site-identifier 1;
                site-preference primary;
                interface ge-2/0/2.0 {
                    remote-site-id 2;
                }
            }
        }
    }
}
```

From configuration mode, confirm your configuration on Router PE2 by entering the **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.

```
[edit]
user@PE2# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

```
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }

    egress-protection {
        context-identifier 198.51.100.3 {
            protector;
            advertise-mode stub-alias;
        }
        traceoptions {
            file ep size 100m;
            flag all;
        }
    }
}
bgp {
    traceoptions {
        file bgp.log world-readable;
    }
    group ibgp {
        type internal;
        local-address 10.255.183.57;
        family inet {
            unicast;
        }
        family l2vpn {
            signaling {
                egress-protection;
            }
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.4;
    }
}
isis {
    traceoptions {
        file isis-edge size 10m world-readable;
        flag error;
    }
    level 1 disable;
    level 2 wide-metrics-only;
    interface all {
        point-to-point;
        level 2 metric 10;
    }
    interface fxp0.0 {
        disable;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}

[edit]
user@PE2# show policy-options
policy-statement lb {
    then {
        load-balance per-packet;
    }
}

[edit]
user@PE2# show routing-options
traceoptions {
    file ro.log;
    flag normal;
    flag route;
}
autonomous-system 100;
forwarding-table {
    export lb;
}

[edit]
user@PE2# show routing-instances
foo {
    instance-type l2vpn;
    egress-protection {
        protector;
    }
    interface ge-2/0/2.0;
    route-distinguisher 10.255.183.57:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo {
                hot-standby;
                site-identifier 1;
                site-preference backup;
                interface ge-2/0/2.0 {
                    remote-site-id 2;
                }
            }
        }
    }
}
```

From configuration mode, confirm your configuration on Router PE3 by entering the **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.

```
[edit]
user@PE3# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    traceoptions {
        file bgp.log world-readable;
    }
    group ibgp {
        type internal;
        local-address 10.255.183.61;
        family inet {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.4;
    }
}
isis {
    traceoptions {
        file isis-edge size 10m world-readable;
        flag error;
    }
    level 1 disable;
    level 2 wide-metrics-only;
    interface all {
        point-to-point;
        level 2 metric 10;
    }
    interface fxp0.0 {
        disable;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}

[edit]
user@PE3# show policy-options
policy-statement lb {
    then {
        load-balance per-packet;
    }
}

[edit]
user@PE3# show routing-options
traceoptions {
    file ro.log;
    flag normal;
    flag route;
}
autonomous-system 100;
forwarding-table {
    export lb;
}

[edit]
user@PE3# show routing-instances
foo {
    instance-type l2vpn;
    interface ge-2/1/2.0;
    route-distinguisher 10.255.183.61:1;
    vrf-target target:9000:1;
    protocols {
        l2vpn {
            encapsulation-type ethernet-vlan;
            site foo {
                site-identifier 2;
                interface ge-2/1/2.0 {
                    remote-site-id 1;
                }
            }
        }
    }
}
```

Verification

IN THIS SECTION

- Verifying the L2VPN Configuration | **539**
- Verifying the Routing Instance Details | **540**
- Verifying the IS-IS Configuration | **541**
- Verifying the MPLS Configuration | **541**

Confirm that the configuration is working properly.

Verifying the L2VPN Configuration

Purpose

Verify that LSP is protected by the connection protection logic.

Action

From operational mode, run the `show l2vpn connections extensive` command.

```
user@PE2> show l2vpn connections extensive
```

```


Layer-2 VPN connections:
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch
Legend for interface status
Up -- operational
Dn -- down
Instance: foo
Local site: foo (1)
```

```
    connection-site           Type  St  Time last up          # Up trans
     2                         rmt   Up  Aug  3 00:08:14 2001           1
       Local circuit: ge-2/0/2.0, Status: Up
       Remote PE: 192.0.2.3
       Incoming label: 32769, Outgoing label: 32768
       Egress Protection: Yes
         Time                  Event            Interface/Lbl/PE
         Aug  3 00:08:14 2001  PE route up
         Aug  3 00:08:14 2001  Out lbl Update               32768
         Aug  3 00:08:14 2001  In lbl Update                32769
         Aug  3 00:08:14 2001  ckt0 up                 fe-0/0/0.0
```

Meaning

The `Egress Protection: Yes` output shows that the given PVC is protected by connection protection logic.

Verifying the Routing Instance Details

Purpose

Verify the routing instance information and the context identifier configured on the primary, which is used as the next-hop address in case of node-link failure.

Action

From operational mode, run the `show route foo detail` command.

```
user@PE2> show route foo detail
```

```
foo:
  Router ID: 0.0.0.0
  Type: l2vpn non-forwarding State: Active
  Interfaces:
    lt-1/2/0.56
  Route-distinguisher: 10.255.255.11:1
  Vrf-import: [ __vrf-import-foo-internal__ ]
  Vrf-export: [ __vrf-export-foo-internal__ ]
  Vrf-import-target: [ target:100:200 ]
  Vrf-export-target: [ target:100:200 ]
  Fast-reroute-priority: low
  Vrf-edge-protection-id:  198.51.100.3
  Tables:
    foo.l2vpn.0           : 5 routes (3 active, 0 holddown, 0 hidden)
    foo.l2id.0            : 6 routes (2 active, 0 holddown, 0 hidden)
```

Meaning

The context-id is set to `198.51.100.3` and the `Vrf-import: [ __vrf-import-foo-internal__]` in the output mentions the policy used for rewriting the next-hop address.

Verifying the IS-IS Configuration

Purpose

Verify the IS-IS context identifier information.

Action

From operational mode, run the `show isis context-identifier detail` command.

```
user@PE2> show isis context-identifier detail
```

```
IS-IS context database:
Context               L  Owner     Role       Primary      Metric
198.51.100.3             2  MPLS      Protector  pro17-b-lr-R1 0
  Advertiser pro17-b, Router ID 10.255.107.49, Level 2, tlv protector
  Advertiser pro17-b-lr-R1, Router ID 10.255.255.11, Metric 1, Level 2, tlv prefix
```

Meaning

Router PE2 is the protector and the configured context identifier is in use for the MPLS protocol.

Verifying the MPLS Configuration

Purpose

Verify the context identifier details on the primary and protector PEs.

Action

From operational mode, run the `show mpls context-identifier detail` command.

```
user@PE1> show mpls context-identifier detail
```

```
ID: 198.51.100.3
  Type: primary, Metric: 1, Mode: alias

Total 1, Primary 1, Protector 0
```

```
user@PE2> show mpls context-identifier detail
```

```
ID: 198.51.100.3
  Type: protector, Metric: 16777215, Mode: alias
  Context table: __198.51.100.3__.mpls.0, Label out: 299968
```

```
user@PE2> show mpls egress-protection detail
```

```


Instance                  Type      Protection-Type
foo                     local-l2vpn Protector
  Route Target 100:200
```

Meaning

Context-id is `198.51.100.3`, advertise-mode is `alias`, the MPLS table created for egress protection is `__198.51.100.3__.mpls.0`, and the egress instance name is `foo`, which is of type `local-l2vpn`.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	Starting in Junos OS Release 14.2, Junos OS supports the restoration of egress traffic when there is a link or node failure in the egress PE node.

Understanding Multisegment Pseudowire for FEC 129

IN THIS SECTION

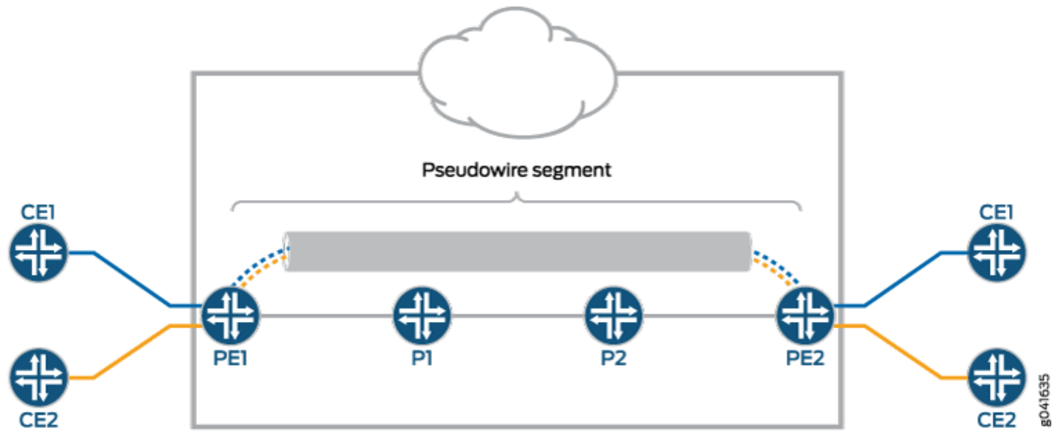
- Understanding Multisegment Pseudowire | 543
- Using FEC 129 for Multisegment Pseudowire | 545
- Establishing a Multisegment Pseudowire Overview | 546
- Pseudowire Status Support for Multisegment Pseudowire | 546
- Pseudowire TLV Support for MS-PW | 547
- Supported and Unsupported Features | 547

Understanding Multisegment Pseudowire

A pseudowire is a Layer 2 circuit or service that emulates the essential attributes of a telecommunications service, such as a T1 line, over an MPLS packet-switched network (PSN). The pseudowire is intended to provide only the minimum necessary functionality to emulate the wire with the required resiliency requirements for the given service definition.

When a pseudowire originates and terminates on the edge of the same PSN, the pseudowire label is unchanged between the originating and terminating provider edge (T-PE) devices. This is called a single-segment pseudowire (SS-PW). Figure 45 on page 544 illustrates an SS-PW established between two PE routers. The pseudowires between the PE1 and PE2 routers are located within the same autonomous system (AS).

Figure 45: L2VPN Pseudowire

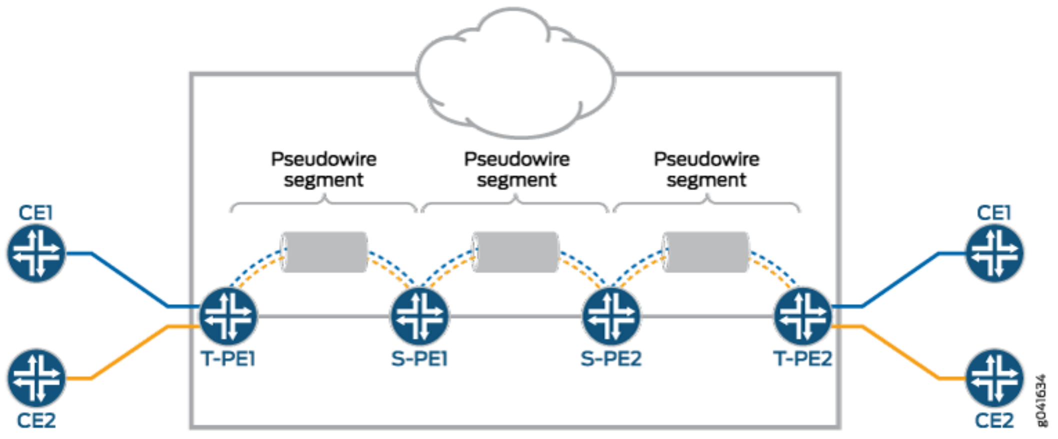


In cases where it is impossible to establish a single pseudowire from a local to a remote PE, either because it is unfeasible or undesirable to establish a single control plane between the two PEs, a multisegment pseudowire (MS-PW) is used.

An MS-PW is a set of two or more contiguous SS-PWs that are made to function as a single point-to-point pseudowire. It is also known as switched pseudowire. MS-PWs can go across different regions or network domains. A region can be considered as an interior gateway protocol (IGP) area or a BGP autonomous system that belongs to the same or different administrative domain. An MS-PW spans multiple cores or ASs of the same or different carrier networks. A Layer 2 VPN MS-PW can include up to 254 pseudowire segments.

Figure 46 on page 545 illustrates a set of two or more pseudowire segments that function as a single pseudowire. The end routers are called terminating PE (T-PE) routers, and the switching routers are called switching PE (S-PE) routers. The S-PE router terminates the tunnels of the preceding and succeeding pseudowire segments in an MS-PW. The S-PE router can switch the control and data planes of the preceding and succeeding pseudowire segments of the MS-PW. An MS-PW is declared to be up when all the single-segment pseudowires are up.

Figure 46: Multisegment Pseudowire



Using FEC 129 for Multisegment Pseudowire

Currently, there are two types of attachment circuit identifiers (AIIs) defined under FEC 129:

- Type 1 AII
- Type 2 AII

The support of an MS-PW for FEC 129 uses type 2 AII. A type 2 AII is globally unique by definition of RFC 5003.

Single-segment pseudowires (SS-PWs) using FEC 129 on an MPLS PSN can use both type 1 and type 2 AII. For an MS-PW using FEC 129, a pseudowire itself is identified as a pair of endpoints. This requires that the pseudowire endpoints be uniquely identified.

In the case of a dynamically placed MS-PW, there is a requirement for the identifiers of attachment circuits to be globally unique, for the purposes of reachability and manageability of the pseudowire. Thus, individual globally unique addresses are allocated to all the attachment circuits and S-PEs that make up an MS-PW.

Type 2 AII is composed of three fields:

- Global_ID—Global identification, which is usually the AS number.
- Prefix—IPv4 address, which is usually the router ID.
- AC_ID—Local attachment circuit, which is a user-configurable value.

Since type 2 AII already contains the T-PE's IP address and it is globally unique, from the FEC 129 pseudowire signaling point of view, the combination (AGI, SAII, TAII) uniquely identifies an MS-PW across all interconnected pseudowire domains.

Establishing a Multisegment Pseudowire Overview

An MS-PW is established by dynamically and automatically selecting the predefined S-PEs and placing the MS-PW between two T-PE devices.

When S-PEs are dynamically selected, each S-PE is automatically discovered and selected using the BGP autodiscovery feature, without the requirement of provisioning the FEC 129 pseudowire-related information on all the S-PEs. BGP is used to propagate pseudowire address information throughout the PSN.

Since there is no manual provisioning of FEC 129 pseudowire information on the S-PEs, the Attachment Group Identifier (AGI) and Attachment Individual Identifier (AII) are reused automatically, and choosing the same set of S-PEs for the pseudowire in both the forwarding and reverse direction is achieved through the active and passive role of each T-PE device.

- Active—The T-PE initiates an LDP label mapping message.
- Passive—The T-PE does not initiate an LDP label mapping message until it receives a label mapping message initiated by the active T-PE. The passive T-PE sends its label mapping message to the same S-PE from where it received the label mapping message originated from its active T-PE. This ensures that the same set of S-PEs are used in the reverse direction.

Pseudowire Status Support for Multisegment Pseudowire

Pseudowire Status Behavior on T-PE

The following pseudowire status messages are relevant on the T-PE:

- 0x00000010—Local PSN-facing pseudowire (egress) transmit fault.
- 0x00000001—Generic nonforwarding fault code. This is set as the local fault code. The local fault code is set at the local T-PE, and LDP sends a pseudowire status TLV message with the same fault code to the remote T-PE.
- Fault codes are bit-wise OR'ed and stored as remote pseudowire status codes.

Pseudowire Status Behavior on S-PE

The S-PE initiates the pseudowire status messages that indicate the pseudowire faults. The SP-PE in the pseudowire notification message hints where the fault was originated.

- When a local fault is detected by the S-PE, a pseudowire status message is sent in both directions along the pseudowire. Since there are no attachment circuits on an S-PE, only the following status messages are relevant:

  - 0x00000008—Local PSN-facing pseudowire (ingress) receive fault.
  - 0x00000010—Local PSN-facing pseudowire (egress) transmit fault.

- To indicate which SS-PW is at fault, an LDP SP-PE TLV is attached with the pseudowire status code in the LDP notification message. The pseudowire status is passed along from one pseudowire to another unchanged by the control plane switching function.
- If an S-PE initiates a pseudowire status notification message with one particular pseudowire status bit, then for the pseudowire status code an S-PE receives, the same bit is processed locally and not forwarded until the S-PE's original status error is cleared.
- An S-PE keeps only two pseudowire status codes for each SS-PW it is involved in – local pseudowire status code and remote pseudowire status code. The value of the remote pseudowire status code is the result of logic or operation of the pseudowire status codes in the chain of SS-PWs preceding this segment. This status code is incrementally updated by each S-PE upon receipt and communicated to the next S-PE. The local pseudowire status is generated locally based on its local pseudowire status.
- Only transmit fault is detected at the SP-PE. When there is no MPLS LSP to reach the next segment, a local transmit fault is detected. The transmit fault is sent to the next downstream segment, and the receive fault is sent to the upstream segment.
- Remote failures received on an S-PE are just passed along the MS-PW unchanged. Local failures are sent to both segments of the pseudowire that the S-PE is involved in.

Pseudowire TLV Support for MS-PW

MS-PW provides the following support for the LDP SP-PE TLV [RFC 6073]:

- The LDP SP-PE TLVs for an MS-PW include:
  - Local IP address
  - Remote IP address
- An SP-PE adds the LDP SP-PE TLV to the label mapping message. Each SP-PE appends the local LDP SP-PE TLV to the SP-PE list it received from the other segment.
- The pseudowire status notification message includes the LDP SP-PE TLV when the notification is generated at the SP-PE.

Supported and Unsupported Features

Junos OS supports the following features with MS-PW:

- MPLS PSN for each SS-PW that builds up the MS-PW.

- The same pseudowire encapsulation for each SS-PW in an MS-PW – Ethernet or VLAN-CCC.
- The generalized PWid FEC with T-LDP as an end-to-end pseudowire signaling protocol to set up each SS-PW.
- MP-BGP to autodiscover the two endpoint PEs for each SS-PW associated with the MS-PW.
- Standard MPLS operation to stitch two side-by-side SS-PWs to form an MS-PW.
- Automatic discovery of S-PE so that the MS-PW can be dynamically placed.
- Minimum provisioning of S-PE.
- Operation, administration, and maintenance (OAM) mechanisms, including end-to-end MPLS ping or end-to-any-S-PE MPLS ping, MPLS path trace, end-to-end VCCV, and Bidirectional Forwarding Detection (BFD).
- Pseudowire swithing point (SP) PE TLV for the MS-PW.
- Composite next hop on MS-PW.
- Pseudowire status TLV for MS-PW.

Junos OS does not support the following MS-PW functionality:

- Mix of LDP FEC 128 and LDP FEC 129.
- Static pseudowire where each label is provisioned staticially.
- Graceful Routing Engine switchover.
- Nonstop active routing.
- Multihoming.
- Partial connectivity verification (originating from an S-PE) in OAM.

Example: Configuring a Multisegment Pseudowire

IN THIS SECTION

- Requirements | 549
- Overview | 549

- Configuration | 556
- Verification | 584
- Troubleshooting | 595

This example shows how to configure a dynamic multisegment pseudowire (MS-PW), where the stitching provider edge (S-PE) devices are automatically and dynamically discovered by BGP, and pseudowires are signaled by LDP using FEC 129. This arrangement requires minimum provisioning on the S-PEs, thereby reducing the configuration burden that is associated with statically configured Layer 2 circuits while still using LDP as the underlying signaling protocol.

Requirements

This example uses the following hardware and software components:

- Six routers that can be a combination of M Series Multiservice Edge Routers, MX Series 5G Universal Routing Platforms, T Series Core Routers, or PTX Series Packet Transport Routers.
  - Two remote PE devices configured as terminating PEs (T-PEs).
  - Two S-PEs configured as:
    - Route reflectors, in the case of interarea configuration.
    - AS boundary routers or route reflectors, in the case of inter-AS configuration.
- Junos OS Release 13.3 or later running on all the devices.

Before you begin:

1. Configure the device interfaces.
2. Configure OSPF or any other IGP protocol.
3. Configure BGP.
4. Configure LDP.
5. Configure MPLS.

Overview

Starting with Junos OS Release 13.3, you can configure an MS-PW using FEC 129 with LDP signaling and BGP autodiscovery in an MPLS packet-switched network (PSN). The MS-PW feature also provides operation, administration, and management (OAM) capabilities, such as ping, traceroute, and BFD, from the T-PE devices.

To enable autodiscovery of S-PEs in an MS-PW, include the `auto-discovery-mspw` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level.

```
family l2vpn {
    auto-discovery-mspw;
}
```

The automatic selection of S-PE and dynamic setting up of an MS-PW rely heavily on BGP. BGP network layer reachability information (NLRI) constructed for the FEC 129 pseudowire to autodiscover the S-PE is called an MS-PW NLRI [draft-ietf-pwe3-dynamic-ms-pw-15.txt]. The MS-PW NLRI is essentially a prefix consisting of a route distinguisher (RD) and FEC 129 source attachment identifier (SAII). It is referred to as a BGP autodiscovery (BGP-AD) route and is encoded as `RD:SAII`.

Only T-PEs that are provisioned with type 2 AIIs initiate their own MS-PW NLRI respectively. Since a type 2 AII is globally unique, an MS-PW NLRI is used to identify a PE device to which the type 2 AII is provisioned. The difference between a type 1 AII and a type 2 AII requires that a new address family indicator (AFI) and subsequent address family identifier (SAFI) be defined in BGP to support an MS-PW. The proposed AFI and SAFI value pair used to identify the MS-PW NLRI is 25 and 6, respectively (pending IANA allocation).

The AFI and SAFI values support autodiscovery of S-PEs and should be configured on both T-PEs that originate the routes, and the S-PEs that participate in the signaling.

Figure 47 on page 551 illustrates an inter-area MS-PW setup between two remote PE routers—T-PE1 and T-PE2. The Provider (P) routers are P1 and P2, and the S-PE routers are S-PE1 and S-PE2. The MS-PW is established between T-PE1 and T-PE2, and all the devices belong to the same AS—AS 100. Since S-PE1 and S-PE2 belong to the same AS, they act as route reflectors and are also known as RR 1 and RR 2, respectively.

Figure 48 on page 551 illustrates an inter-AS MS-PW setup. The MS-PW is established between T-PE1 and T-PE2, where T-PE1, P1, and S-PE1 belong to AS 1, and S-PE2, P2, and T-PE2 belong to AS 2. Since S-PE1 and S-PE2 belong to different ASs, they are configured as ASBR routers and are also known as ASBR 1 and ASBR 2, respectively.

Figure 47: Interarea Multisegment Pseudowire

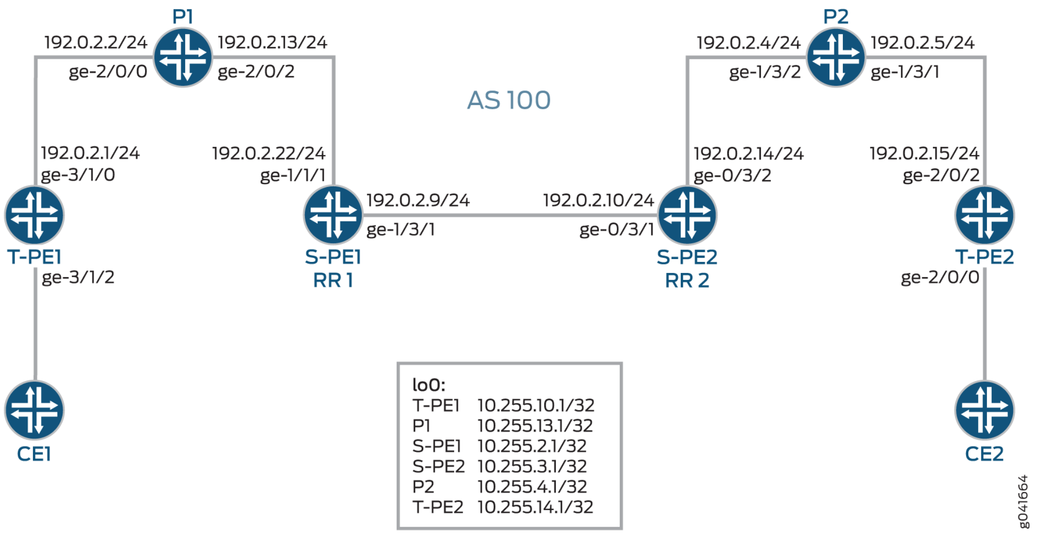
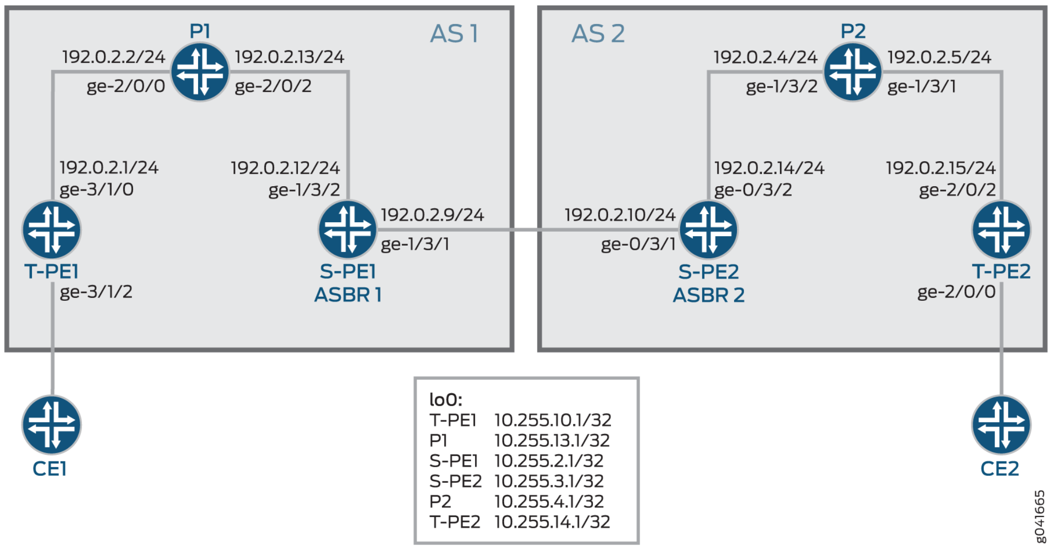


Figure 48: Inter-AS Multisegment Pseudowire



The following sections provide information about how an MS-PW is established in an interarea and inter-AS scenario.

Minimum Configuration Requirements on S-PE

In order to dynamically discover both ends of an SS-PW and set up a T-LDP session dynamically, the following is required:

- For interarea MS-PW, each S-PE plays both an ABR and BGP route reflector role.

  In the interarea case, as seen in Figure 47 on page 551, the S-PE plays a BGP route reflector role and reflects the BGP-AD route to its client. A BGP-AD route advertised by one T-PE eventually reaches its remote T-PE. Because of the next-hop-self set by each S-PE, the S-PE or T-PE that receives a BGP-AD route can always discover the S-PE that advertises the BGP-AD in its local AS or local area through the BGP next hop.

- For inter-AS MS-PW, each S-PE plays either an ASBR or a BGP route reflector role.

  In an MS-PW, the two T-PEs initiate a BGP-AD route respectively. When the S-PE receives the BGP-AD route through either the IBGP session with the T-PE or through a regular BGP-RR, it sets the next-hop-self before re-advertising the BGP-AD route to one or more of its EBGP peers in the inter-AS case, as seen in Figure 48 on page 551.

- Each S-PE must set next-hop-self when re-advertising or reflecting a BGP-AD route for the MS-PW.

Active and Passive Role of T-PE

To ensure that the same set of S-PEs are being used for a MS-PW in both directions, the two T-PEs play different roles in terms of FEC 129 signaling. This is to avoid different paths being chosen by T-PE1 and T-PE2 when each S-PE is dynamically selected for an MS-PW.

When an MS-PW is signaled using FEC 129, each T-PE might independently start signaling the MS-PW. The signaling procedure can result in an attempt to set up each direction of the MS-PW through different S-PEs.

To avoid this situation, one of the T-PEs must start the pseudowire signaling (active role), while the other waits to receive the LDP label mapping before sending the respective pseudowire LDP label mapping message (passive role). When the MS-PW path is dynamically placed, the active T-PE (the Source T-PE) and the passive T-PE (the Target T-PE) must be identified before signaling is initiated for a given MS-PW. The determination of which T-PE assumes the active role is done based on the SAII value, where the T-PE that has a larger SAII value plays the active role.

In this example, the SAII values of T-PE1 and T-PE 2 are `800:800:800` and `700:700:700`, respectively. Since T-PE1 has a higher SAII value, it assumes the active role and T-PE2 assumes the passive role.

Directions for Establishing an MS-PW

The directions used by the S-PE for setting up the MS-PW are:

- Forwarding direction—From an active T-PE to a passive T-PE.

  In this direction, the S-PEs perform a BGP-AD route lookup to determine the next-hop S-PE to send the label mapping message.

- Reverse direction—From a passive T-PE to an active T-PE.

In this direction, the S-PEs do not perform a BGP-AD route lookup, because the label mapping messages are received from the T-PEs, and the stitching routes are installed in the S-PEs.

In this example, the MS-PW is established in the forwarding direction from T-PE1 to T-PE2. When the MS-PW is placed from T-PE2 to T-PE1, the MS-PW is established in the reverse direction.

Autodiscovery and Dynamic Selection of S-PE

A new AFI and SAFI value is defined in BGP to support the MS-PWs based on type 2 AII. This new address family supports autodiscovery of S-PEs. This address family must be configured on both the TPEs and SPEs.

It is the responsibility of the Layer 2 VPN component to dynamically select the next S-PE to use along the MS-PW in the forwarding direction.

- In the forwarding direction, the selection of the next S-PE is based on the BGP-AD route advertised by the BGP and pseudowire FEC information sent by the LDP. The BGP-AD route is initiated by the passive T-PE (T-PE2) in the reverse direction while the pseudowire FEC information is sent by LDP from the active T-PE (T-PE1) in the forwarding direction.

- In the reverse direction, the next S-PE (S-PE2) or the active T-PE (T-PE1) is obtained by looking up the S-PE (S-PE1) that it used to set up the pseudowire in the forwarding direction.

Provisioning a T-PE

To support FEC 129 type 2 AII, the T-PE needs to configure its remote T-PE's IP address, a global ID, and an attachment circuit ID. Explicit paths where a set of S-PEs to use is explicitly specified on a T-PE is not supported. This eliminates the need to provision each S-PE with a type 2 AII.

Stitching an MS-PW

An S-PE performs the following MPLS label operations before forwarding the received label mapping message to the next S-PE:

1. Pops the MPLS tunnel label.

2. Pops the VC label.

3. Pushes a new VC label.

4. Pushes an MPLS tunnel label used for the next segment.

Establishing an MS-PW

After completing the necessary configuration, an MS-PW is established in the following manner:

1. The SAII values are exchanged between T-PE1 and T-PE2 using BGP.

T-PE1 assumes the active T-PE role, because it is configured with a higher SAII value. T-PE2 becomes the passive T-PE.

2. T-PE1 receives the BGP-AD route originated by T-PE2. It compares the AII values obtained from T-PE2 in the received BGP-AD route against the AII values provisioned locally.

3. If the AII values match, T-PE1 performs a BGP-AD route lookup to elect the first S-PE (S-PE1).

4. T-PE1 sends an LDP label mapping message to S-PE1.

5. Using the BGP-AD route originated from T-PE2, and the LDP label mapping message received from T-PE1, S-PE1 selects the next S-PE (S-PE2) in the forwarding direction.

   To do this, S-PE1 compares SAII obtained from the BGP-AD route against the TAI from the LDP label mapping message.

6. If the AII values match, S-PE1 finds S-PE2 through the BGP next hop associated with the BGP-AD route.

7. The process of selecting S-PE goes on until the last S-PE establishes a T-LDP session with T-PE2. When T-PE2 receives the LDP label mapping message from the last S-PE (S-PE2), it initiates its own label mapping message and sends it back to S-PE2.

8. When all the label mapping messages are received on S-PE1 and S-PE2, the S-PEs install the stitching routes. Thus, when the MS-PW is established in the reverse direction, the S-PEs need not perform BGP-AD route lookup to determine its next hop as it did in the forwarding direction.

OAM Support for an MS-PW

After the MS-PW is established, the following OAM capabilities can be executed from the T-PE devices:

- Ping

  - End-to-End Connectivity Verification Between T-PEs

    If T-PE1, S-PEs, and T-PE2 support Control Word (CW), the pseudowire control plane automatically negotiates the use of the CW. Virtual Circuit Connectivity Verification (VCCV) Control Channel (CC) Type 3 will function correctly whether or not the CW is enabled on the pseudowire. However, VCCV Type 1, which is used for end-to-end verification only, is only supported if the CW is enabled.

    The following is a sample:

    Ping from T-P1 to T-PE2

    ```
    user@T-PE1> ping mpls l2vpn fec129 instance instance-name local-id SAII of T-PE1 remote-pe-
    address address of T-PE2 remote-id TAII of T-PE2
    ```

or

```
user@T-PE1> ping mpls l2vpn fec129 interface CE1-facing interface
```

- Partial Connectivity Verification from T-PE to Any S-PE

  To trace part of an MS-PW, the TTL of the pseudowire label can be used to force the VCCV message to pop out at an intermediate node. When the TTL expires, the S-PE can determine that the packet is a VCCV packet either by checking the CW or by checking for a valid IP header with UDP destination port 3502 (if the CW is not in use). The packet should then be diverted to VCCV processing.

  If T-PE1 sends a VCCV message with the TTL of the pseudowire label equal to 1, the TTL expires at the S-PE. T-PE1 can thus verify the first segment of the pseudowire.

  The VCCV packet is built according to RFC 4379. All the information necessary to build the VCCV LSP ping packet is collected by inspecting the S-PE TLVs. This use of the TTL is subject to the caution expressed in RFC 5085. If a penultimate LSR between S-PEs or between an S-PE and a T-PE manipulates the pseudowire label TTL, the VCCV message might not emerge from the MS-PW at the correct S-PE.

  The following is a sample:

  Ping from T-PE1 to S-PE

  ```
  user@T-PE1> ping mpls l2vpn fec129 interface CE1-facing interface bottom-label-ttl segment
  ```

  The `bottom-label-ttl` value is 1 for S-PE1 and 2 for S-PE2.

  The `bottom-label-ttl` statement sets the correct VC label TTL, so the packets are popped to the correct SS-PW for VCCV processing.

  NOTE: Junos OS supports VCCV Type 1 and Type 3 for the MS-PW OAM capability. VCCV Type 2 is not supported.

- Traceroute

Traceroute tests each S-PE along the path of the MS-PW in a single operation similar to LSP trace. This operation is able to determine the actual data path of the MS-PW, and is used for dynamically signaled MS-PWs.

```
user@T-PE1> traceroute mpls l2vpn fec129 interface CE1-facing interface
```

- Bidirectional Forwarding Detection

  Bidirectional Forwarding Detection (BFD) is a detection protocol designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. In addition to fast forwarding path failure detection, BFD provides a consistent failure detection method for network administrators. The router or switch can be configured to log a system log (syslog) message when BFD goes down.

```
user@T-PE1> show bfd session extensive
```

Configuration

IN THIS SECTION

- Configuring an Interarea MS-PW | 556
- Configuring an Inter-AS MS-PW | 570

Configuring an Interarea MS-PW

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.

T-PE1

```
set interfaces ge-3/1/0 unit 0 family inet address 192.0.2.1/24
set interfaces ge-3/1/0 unit 0 family mpls
set interfaces ge-3/1/2 encapsulation ethernet-ccc
set interfaces ge-3/1/2 unit 0
set interfaces lo0 unit 0 family inet address 10.255.10.1/32 primary
set routing-options autonomous-system 100
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.10.1
set protocols bgp group mspw neighbor 10.255.2.1
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances ms-pw instance-type l2vpn
set routing-instances ms-pw interface ge-3/1/2.0
set routing-instances ms-pw route-distinguisher 10.10.10.10:15
set routing-instances ms-pw l2vpn-id l2vpn-id:100:15
set routing-instances ms-pw vrf-target target:100:115
set routing-instances ms-pw protocols l2vpn site CE1 source-attachment-identifier 800:800:800
set routing-instances ms-pw protocols l2vpn site CE1 interface ge-3/1/2.0 target-attachment-
identifier 700:700:700
set routing-instances ms-pw protocols l2vpn pseudowire-status-tlv
set routing-instances ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 300
```

P1

```
set interfaces ge-2/0/0 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/0 unit 0 family mpls
set interfaces ge-2/0/2 unit 0 family inet address 192.0.2.13/24
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.13.1/32 primary
set routing-options autonomous-system 100
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
```

S-PE1 (RR 1)

```
set interfaces ge-1/3/1 unit 0 family inet address 192.0.2.9/24
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/2 unit 0 family inet address 192.0.2.22/24
set interfaces ge-1/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.2.1/32 primary
set routing-options autonomous-system 100
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.2.1
set protocols bgp group mspw export next-hop-self
set protocols bgp group mspw cluster 203.0.113.0
set protocols bgp group mspw neighbor 10.255.10.1
set protocols bgp group mspw neighbor 10.255.3.1
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set policy-options policy-statement next-hop-self then next-hop self
set policy-options policy-statement send-inet0 from protocol bgp
set policy-options policy-statement send-inet0 then accept
```

S-PE2 (RR 2)

```
set interfaces ge-0/3/1 unit 0 family inet address 192.0.2.10/24
set interfaces ge-0/3/1 unit 0 family mpls
set interfaces ge-0/3/2 unit 0 family inet address 192.0.2.14/24
set interfaces ge-0/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.3.1/32 primary
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.3.1
set protocols bgp group mspw export next-hop-self
set protocols bgp group mspw cluster 198.51.100.0
set protocols bgp group mspw neighbor 10.255.2.1
set protocols bgp group mspw neighbor 10.255.14.1
set protocols bgp group int type internal
set protocols bgp group int local-address 10.255.3.1
set protocols bgp group int neighbor 10.255.2.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set policy-options policy-statement next-hop-self then next-hop self
set policy-options policy-statement send-inet0 from protocol bgp
set policy-options policy-statement send-inet0 then accept
```

P2

```
set interfaces ge-1/3/1 unit 0 family inet address 192.0.2.5/24
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/2 unit 0 family inet address 192.0.2.4/24
set interfaces ge-1/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.4.1/32 primary
set routing-options autonomous-system 100
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
```

T-PE2

```
set interfaces ge-2/0/0 encapsulation ethernet-ccc
set interfaces ge-2/0/0 unit 0
set interfaces ge-2/0/2 unit 0 family inet address 192.0.2.15/24
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.1/32 primary
set routing-options autonomous-system 100
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.14.1
set protocols bgp group mspw neighbor 10.255.3.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances ms-pw instance-type l2vpn
set routing-instances ms-pw interface ge-2/0/0.0
set routing-instances ms-pw route-distinguisher 10.10.10.10:15
set routing-instances ms-pw l2vpn-id l2vpn-id:100:15
set routing-instances ms-pw vrf-target target:100:115
set routing-instances ms-pw protocols l2vpn site CE2 source-attachment-identifier 700:700:700
set routing-instances ms-pw protocols l2vpn site CE2 interface ge-2/0/0.0 target-attachment-
identifier 800:800:800
set routing-instances ms-pw protocols l2vpn pseudowire-status-tlv
set routing-instances ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 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*.

To configure T-PE1 in the interarea scenario:

NOTE: Repeat this procedure for the T-PE2 device in the MPLS domain, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure the T-PE1 interfaces.

```
[edit interfaces]
user@T-PE1# set ge-3/1/0 unit 0 family inet address 192.0.2.1/24
user@T-PE1# set ge-3/1/0 unit 0 family mpls
user@T-PE1# set ge-3/1/2 encapsulation ethernet-ccc
user@T-PE1# set ge-3/1/2 unit 0
user@T-PE1# set lo0 unit 0 family inet address 10.255.10.1/32 primary
```

2. Set the autonomous system number.

```
[edit routing-options]
user@T-PE1# set autonomous-system 100
```

3. Enable MPLS on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set mpls interface all
user@T-PE1# set mpls interface fxp0.0 disable
```

4. Enable autodiscovery of intermediate S-PEs that make up the MS-PW using BGP.

```
[edit protocols]
user@T-PE1# set bgp family l2vpn auto-discovery-mspw
```

5. Configure the BGP group for T-PE1.

```
[edit protocols]
user@T-PE1# set bgp group mspw type internal
```

6. Assign local and neighbor addresses to the mspw group for T-PE1 to peer with S-PE1.

```
[edit protocols]
user@T-PE1# set bgp group mspw local-address 10.255.10.1
user@T-PE1# set bgp group mspw neighbor 10.255.2.1
```

7. Configure OSPF on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set ospf area 0.0.0.0 interface lo0.0
user@T-PE1# set ospf area 0.0.0.0 interface all
user@T-PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

8. Configure LDP on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set ldp interface all
user@T-PE1# set ldp interface fxp0.0 disable
user@T-PE1# set ldp interface lo0.0
```

9. Configure the Layer 2 VPN routing instance on T-PE1.

```
[edit routing-instances]
user@T-PE1# set ms-pw instance-type l2vpn
```

10. Assign the interface name for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw interface ge-3/1/2.0
```

11. Configure the route distinguisher for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw route-distinguisher 10.10.10.10:15
```

12. Configure the Layer 2 VPN ID community for FEC 129 MS-PW.

```
[edit routing-instances]
user@T-PE1# set ms-pw l2vpn-id l2vpn-id:100:15
```

13. Configure a VPN routing and forwarding (VRF) target for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw vrf-target target:100:115
```

14. Configure the source attachment identifier (SAI) value using Layer 2 VPN as the routing protocol for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn site CE1 source-attachment-identifier 800:800:800
```

15. Assign the interface name that connects the CE1 site to the VPN, and configure the target attachment identifier (TAI) value using Layer 2 VPN as the routing protocol for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn site CE1 interface ge-3/1/2.0 target-attachment-
identifier 700:700:700
```

16. (Optional) Configure T-PE1 to send MS-PW status TLVs.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn pseudowire-status-tlv
```

17. (Optional) Configure OAM capabilities for the VPN.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 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*.

To configure S-PE1 (RR 1) in the interarea scenario:

NOTE: Repeat this procedure for the S-PE2 (RR 2) device in the MPLS domain, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure the S-PE1 interfaces.

```
[edit interfaces]
user@S-PE1# set ge-1/3/1 unit 0 family inet address 192.0.2.9/24
user@S-PE1# set ge-1/3/1 unit 0 family mpls
user@S-PE1# set ge-1/3/2 unit 0 family inet address 192.0.2.22/24
user@S-PE1# set ge-1/3/2 unit 0 family mpls
user@S-PE1# set lo0 unit 0 family inet address 10.255.2.1/32 primary
```

2. Set the autonomous system number.

```
[edit routing-options]
user@S-PE1# set autonomous-system 100
```

3. Enable MPLS on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@S-PE1# set mpls interface all
user@S-PE1# set mpls interface fxp0.0 disable
```

4. Enable autodiscovery of S-PE using BGP.

```
[edit protocols]
user@S-PE1# set bgp family l2vpn auto-discovery-mspw
```

5. Configure the BGP group for S-PE1.

```
[edit protocols]
user@S-PE1# set bgp group mspw type internal
```

6. Configure S-PE1 to act as a route reflector.

```
[edit protocols]
user@S-PE1# set bgp group mspw export next-hop-self
user@S-PE1# set bgp group mspw cluster 203.0.113.0
```

7. Assign local and neighbor addresses to the mspw group for S-PE1 to peer with T-PE1 and S-PE2.

```
[edit protocols]
user@S-PE1# set bgp group mspw local-address 10.255.2.1
user@S-PE1# set bgp group mspw neighbor 10.255.10.1 (to T-PE1)
user@S-PE1# set bgp group mspw neighbor 10.255.3.1 (to S-PE2)
```

8. Configure OSPF on all the interfaces of S-PE1, excluding the management interface.

```
[edit protocols]
user@S-PE1# set ospf area 0.0.0.0 interface all
user@S-PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
user@S-PE1# set ospf area 0.0.0.0 interface lo0.0
```

9. Configure LDP on all the interfaces of S-PE1, excluding the management interface.

```
[edit protocols]
user@S-PE1# set ldp interface all
user@S-PE1# set ldp interface fxp0.0 disable
user@S-PE1# set ldp interface lo0.0
```

10. Define the policy for enabling next-hop-self and accepting BGP traffic on S-PE1.

```
[edit policy-options]
user@S-PE1# set policy-statement next-hop-self then next-hop self
user@S-PE1# set policy-statement send-inet0 from protocol bgp
user@S-PE1# set policy-statement send-inet0 then accept
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show routing-instances`, `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.

T-PE1

```
user@T-PE1# show interfaces
ge-3/1/0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
        family mpls;
    }
}
ge-3/1/2 {
    encapsulation ethernet-ccc;
    unit 0;
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.10.1/32 {
                primary;
            }
        }
    }
}
```

```
user@T-PE1# show routing-options
autonomous-system 100;
```

```
user@T-PE1# show protocols
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    family l2vpn {
        auto-discovery-mspw;
    }
    group mspw {
        type internal;
        local-address 10.255.10.1;
        neighbor 10.255.2.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@T-PE1# show routing-instances
ms-pw {
    instance-type l2vpn;
    interface ge-3/1/2.0;
    route-distinguisher 10.10.10.10:15;
    l2vpn-id l2vpn-id:100:15;
    vrf-target target:100:115;
    protocols {
        l2vpn {
            site CE1 {
                source-attachment-identifier 800:800:800;
                interface ge-3/1/2.0 {
                    target-attachment-identifier 700:700:700;
                }
            }
            pseudowire-status-tlv;
            oam {
                bfd-liveness-detection {
                    minimum-interval 300;
                }
            }
        }
    }
}
```

S-PE1 (RR 1)

```
user@S-PE1# show interfaces
ge-1/3/1 {
    unit 0 {
        family inet {
            address 192.0.2.9/24;
        }
        family mpls;
    }
}
ge-1/3/2 {
    unit 0 {
        family inet {
            address 192.0.2.22/24;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.2.1/32 {
                primary;
            }
        }
    }
}
```

```
user@S-PE1# show routing-options
autonomous-system 100;
```

```
user@S-PE1# show protocols
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    family l2vpn {
        auto-discovery-mspw;
    }
    group mspw {
        type internal;
        local-address 10.255.2.1;
        export next-hop-self;
        cluster 203.0.113.0;
        neighbor 10.255.10.1;
        neighbor 10.255.3.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0;
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@S-PE1# show policy-options
policy-statement next-hop-self {
    then {
        next-hop self;
    }
}
policy-statement send-inet0 {
    from protocol bgp;
    then accept;
}
```

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

Configuring an Inter-AS MS-PW

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.

T-PE1

```
set interfaces ge-3/1/0 unit 0 family inet address 192.0.2.1/24
set interfaces ge-3/1/0 unit 0 family mpls
set interfaces ge-3/1/2 encapsulation ethernet-ccc
set interfaces ge-3/1/2 unit 0
set interfaces lo0 unit 0 family inet address 10.255.10.1/32 primary
set routing-options autonomous-system 1
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.10.1
set protocols bgp group mspw neighbor 10.255.2.1
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances ms-pw instance-type l2vpn
set routing-instances ms-pw interface ge-3/1/2.0
set routing-instances ms-pw route-distinguisher 10.10.10.10:15
set routing-instances ms-pw l2vpn-id l2vpn-id:100:15
set routing-instances ms-pw vrf-target target:100:115
set routing-instances ms-pw protocols l2vpn site CE1 source-attachment-identifier 800:800:800
set routing-instances ms-pw protocols l2vpn site CE1 interface ge-3/1/2.0 target-attachment-
identifier 700:700:700
```

```
set routing-instances ms-pw protocols l2vpn pseudowire-status-tlv
set routing-instances ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 300
```

P1

```
set interfaces ge-2/0/0 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/0 unit 0 family mpls
set interfaces ge-2/0/2 unit 0 family inet address 192.0.2.13/24
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.13.1/32 primary
set routing-options autonomous-system 1
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
```

S-PE1 (ASBR 1)

```
set interfaces ge-1/3/1 unit 0 family inet address 192.0.2.9/24
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/2 unit 0 family inet address 192.0.2.22/24
set interfaces ge-1/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.2.1/32 primary
set routing-options autonomous-system 1
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group to_T-PE1 type internal
set protocols bgp group to_T-PE1 local-address 10.255.2.1
set protocols bgp group to_T-PE1 export next-hop-self
set protocols bgp group to_T-PE1 neighbor 10.255.10.1
set protocols bgp group to_S-PE2 type external
set protocols bgp group to_S-PE2 local-address 10.255.2.1
set protocols bgp group to_S-PE2 peer-as 2
set protocols bgp group to_S-PE2 neighbor 10.255.3.1 multihop ttl 1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set policy-options policy-statement next-hop-self then next-hop self
```

S-PE2 (ASBR 2)

```
set interfaces ge-0/3/1 unit 0 family inet address 192.0.2.10/24
set interfaces ge-0/3/1 unit 0 family mpls
set interfaces ge-0/3/2 unit 0 family inet address 192.0.2.14/24
set interfaces ge-0/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.3.1/32 primary
set routing-options autonomous-system 2
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group to_T-PE2 type internal
set protocols bgp group to_T-PE2 local-address 10.255.3.1
set protocols bgp group to_T-PE2 export next-hop-self
set protocols bgp group to_T-PE2 neighbor 10.255.14.1
set protocols bgp group to_S-PE1 type external
set protocols bgp group to_S-PE1 local-address 10.255.3.1
set protocols bgp group to_S-PE1 peer-as 1
set protocols bgp group to_S-PE1 neighbor 10.255.2.1 multihop ttl 1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set policy-options policy-statement next-hop-self then next-hop self
```

P2

```
set interfaces ge-1/3/1 unit 0 family inet address 192.0.2.5/24
set interfaces ge-1/3/1 unit 0 family mpls
set interfaces ge-1/3/2 unit 0 family inet address 192.0.2.4/24
set interfaces ge-1/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.4.1/32 primary
set routing-options autonomous-system 2
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
```

T-PE2

```
set interfaces ge-2/0/0 encapsulation ethernet-ccc
set interfaces ge-2/0/0 unit 0
set interfaces ge-2/0/2 unit 0 family inet address 192.0.2.15/24
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.1/32 primary
set routing-options autonomous-system 2
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp family l2vpn auto-discovery-mspw
set protocols bgp group mspw type internal
set protocols bgp group mspw local-address 10.255.14.1
set protocols bgp group mspw neighbor 10.255.3.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances ms-pw instance-type l2vpn
set routing-instances ms-pw interface ge-2/0/0.0
set routing-instances ms-pw route-distinguisher 10.10.10.10:15
set routing-instances ms-pw l2vpn-id l2vpn-id:100:15
set routing-instances ms-pw vrf-target target:100:115
set routing-instances ms-pw protocols l2vpn site CE2 source-attachment-identifier 700:700:700
set routing-instances ms-pw protocols l2vpn site CE2 interface ge-2/0/0.0 target-attachment-
identifier 800:800:800
set routing-instances ms-pw protocols l2vpn pseudowire-status-tlv
set routing-instances ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 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*.

To configure the T-PE1 router in the inter-AS scenario:

NOTE: Repeat this procedure for the T-PE2 device in the MPLS domain, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure the T-PE1 interfaces.

```
[edit interfaces]
user@T-PE1# set ge-3/1/0 unit 0 family inet address 192.0.2.1/24
user@T-PE1# set ge-3/1/0 unit 0 family mpls
user@T-PE1# set ge-3/1/2 encapsulation ethernet-ccc
user@T-PE1# set ge-3/1/2 unit 0
user@T-PE1# set lo0 unit 0 family inet address 10.255.10.1/32 primary
```

2. Set the autonomous system number.

```
[edit routing-options]
user@T-PE1# set autonomous-system 1
```

3. Enable MPLS on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set mpls interface all
user@T-PE1# set mpls interface fxp0.0 disable
```

4. Enable autodiscovery of intermediate S-PEs that make up the MS-PW using BGP.

```
[edit protocols]
user@T-PE1# set bgp family l2vpn auto-discovery-mspw
```

5. Configure the BGP group for T-PE1.

```
[edit protocols]
user@T-PE1# set bgp group mspw type internal
```

6. Assign local and neighbor addresses to the mspw group for T-PE1 to peer with S-PE1.

```
[edit protocols]
user@T-PE1# set bgp group mspw local-address 10.255.10.1
user@T-PE1# set bgp group mspw neighbor 10.255.2.1
```

7. Configure OSPF on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set ospf area 0.0.0.0 interface lo0.0
user@T-PE1# set ospf area 0.0.0.0 interface all
user@T-PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

8. Configure LDP on all the interfaces of T-PE1, excluding the management interface.

```
[edit protocols]
user@T-PE1# set ldp interface all
user@T-PE1# set ldp interface fxp0.0 disable
user@T-PE1# set ldp interface lo0.0
```

9. Configure the Layer 2 VPN routing instance on T-PE1.

```
[edit routing-instances]
user@T-PE1# set ms-pw instance-type l2vpn
```

10. Assign the interface name for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw interface ge-3/1/2.0
```

11. Configure the route distinguisher for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw route-distinguisher 10.10.10.10:15
```

12. Configure the Layer 2 VPN ID community for FEC 129 MS-PW.

```
[edit routing-instances]
user@T-PE1# set ms-pw l2vpn-id l2vpn-id:100:15
```

13. Configure a VPN routing and forwarding (VRF) target for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw vrf-target target:100:115
```

14. Configure the source attachment identifier (SAI) value using Layer 2 VPN as the routing protocol for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn site CE1 source-attachment-identifier 800:800:800
```

15. Assign the interface name that connects the CE1 site to the VPN, and configure the target attachment identifier (TAI) value using Layer 2 VPN as the routing protocol for the mspw routing instance.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn site CE1 interface ge-3/1/2.0 target-attachment-
identifier 700:700:700
```

16. (Optional) Configure T-PE1 to send MS-PW status TLVs.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn pseudowire-status-tlv
```

17. (Optional) Configure OAM capabilities for the VPN.

```
[edit routing-instances]
user@T-PE1# set ms-pw protocols l2vpn oam bfd-liveness-detection minimum-interval 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*.

To configure S-PE1 (ASBR 1) in the inter-AS scenario:

NOTE: Repeat this procedure for the S-PE2 (ASBR 2) device in the MPLS domain, after modifying the appropriate interface names, addresses, and other parameters.

1. Configure S-PE1 (ASBR 1) interfaces.

```
[edit interfaces]
user@S-PE1# set ge-1/3/1 unit 0 family inet address 192.0.2.9/24
user@S-PE1# set ge-1/3/1 unit 0 family mpls
user@S-PE1# set ge-1/3/2 unit 0 family inet address 192.0.2.22/24
user@S-PE1# set ge-1/3/2 unit 0 family mpls
user@S-PE1# set lo0 unit 0 family inet address 10.255.2.1/32 primary
```

2. Set the autonomous system number.

```
[edit routing-options]
user@S-PE1# set autonomous-system 1
```

3. Enable MPLS on all the interfaces of S-PE1 (ASBR 1), excluding the management interface.

```
[edit protocols]
user@S-PE1# set mpls interface all
user@S-PE1# set mpls interface fxp0.0 disable
```

4. Enable autodiscovery of S-PE using BGP.

```
[edit protocols]
user@S-PE1# set bgp family l2vpn auto-discovery-mspw
```

5. Configure the IBGP group for S-PE1 (ASBR 1) to peer with T-PE1.

```
[edit protocols]
user@S-PE1# set bgp group to_T-PE1 type internal
```

6. Configure the IBGP group parameters.

```
[edit protocols]
user@S-PE1# set bgp group to_T-PE1 local-address 10.255.2.1
user@S-PE1# set bgp group to_T-PE1 export next-hop-self
user@S-PE1# set bgp group to_T-PE1 neighbor 10.255.10.1
```

7. Configure the EBGP group for S-PE1 (ASBR 1) to peer with S-PE2 (ASBR 2).

```
[edit protocols]
user@S-PE1# set bgp group to_S-PE2 type external
```

8. Configure the EBGP group parameters.

```
[edit protocols]
user@S-PE1# set bgp group to_S-PE2 local-address 10.255.2.1
user@S-PE1# set bgp group to_S-PE2 peer-as 2
user@S-PE1# set bgp group to_S-PE2 neighbor 10.255.3.1 multihop ttl 1
```

9. Configure OSPF on all the interfaces of S-PE1 (ASBR 1), excluding the management interface.

```
[edit protocols]
user@S-PE1# set ospf area 0.0.0.0 interface all
user@S-PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
user@S-PE1# set ospf area 0.0.0.0 interface lo0.0 passive
```

10. Configure LDP on all the interfaces of S-PE1 (ASBR 1), excluding the management interface.

```
[edit protocols]
user@S-PE1# set ldp interface all
user@S-PE1# set ldp interface fxp0.0 disable
user@S-PE1# set ldp interface lo0.0
```

11. Define the policy for enabling next-hop-self on S-PE1 (ASBR 1).

```
[edit policy-options]
user@S-PE1# set policy-statement next-hop-self then next-hop self
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show routing-instances`, `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.

T-PE1

```
user@T-PE1# show interfaces
ge-3/1/0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
        family mpls;
    }
}
ge-3/1/2 {
    encapsulation ethernet-ccc;
    unit 0;
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.10.1/32 {
                primary;
            }
        }
    }
}
```

```
user@T-PE1# show routing-options
autonomous-system 1;
```

```
user@T-PE1# show protocols
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    family l2vpn {
        auto-discovery-mspw;
    }
    group mspw {
        type internal;
        local-address 10.255.10.1;
        neighbor 10.255.2.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0;
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@T-PE1# show routing-instances
ms-pw {
    instance-type l2vpn;
    interface ge-3/1/2.0;
    route-distinguisher 10.10.10.10:15;
    l2vpn-id l2vpn-id:100:15;
    vrf-target target:100:115;
    protocols {
        l2vpn {
            site CE1 {
                source-attachment-identifier 800:800:800;
                interface ge-3/1/2.0 {
                    target-attachment-identifier 700:700:700;
                }
            }
            pseudowire-status-tlv;
            oam {
                bfd-liveness-detection {
                    minimum-interval 300;
                }
            }
        }
    }
}
```

S-PE1 (RR 1)

```
user@S-PE1# show interfaces
ge-1/3/1 {
    unit 0 {
        family inet {
            address 192.0.2.9/24;
        }
        family mpls;
    }
}
ge-1/3/2 {
    unit 0 {
        family inet {
            address 192.0.2.22/24;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.2.1/32 {
                primary;
            }
        }
    }
}
```

```
user@T-PE1# show routing-options
autonomous-system 1;
```

```
user@S-PE1# show protocols
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    family l2vpn {
        auto-discovery-mspw;
    }
    group to_T-PE1 {
        type internal;
        local-address 10.255.2.1;
        export next-hop-self;
        neighbor 10.255.10.1;
    }
    group to_S-PE2 {
        type external;
        local-address 10.255.2.1;
        peer-as 2;
        neighbor 10.255.3.1 {
            multihop {
                ttl 1;
            }
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@T-PE1# show policy-options
policy-statement next-hop-self {
    then {
        next-hop self;
    }
}
```

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

Verification

IN THIS SECTION

- Verifying the Routes | 584
- Verifying the LDP Database | 587
- Checking the MS-PW Connections on T-PE1 | 588
- Checking the MS-PW Connections on S-PE1 | 590
- Checking the MS-PW Connections on S-PE2 | 592
- Checking the MS-PW Connections on T-PE2 | 593

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 `bgp.l2vpn.1`, `ldp.l2vpn.1`, `mpls.0`, and `ms-pw.l2vpn.1` routing tables.

From operational mode, run the `show route table bgp.l2vpn.1` command.

```
user@T-PE1> show route table bgp.l2vpn.1
bgp.l2vpn.1: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.10.10.10:15:700:0.0.2.188:700/160 AD2
                   *[BGP/170] 16:13:11, localpref 100, from 10.255.2.1
                      AS path: 2 I, validation-state: unverified
                    > to 203.0.113.2 via ge-3/1/0.0, Push 300016
```

From operational mode, run the `show route table ldp.l2vpn.1` command.

```
user@T-PE1> show route table ldp.l2vpn.1
ldp.l2vpn.1: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.255.2.1:CtrlWord:5:100:15:700:0.0.2.188:700:800:0.0.3.32:800/304 PW2
                   *[LDP/9] 16:21:27
                      Discard
```

From operational mode, run the `show route table mpls.0` command.

```
user@T-PE1> show route table mpls.0
mpls.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w6d 00:28:26, metric 1
                      Receive
1                  *[MPLS/0] 1w6d 00:28:26, metric 1
                      Receive
2                  *[MPLS/0] 1w6d 00:28:26, metric 1
                      Receive
13                 *[MPLS/0] 1w6d 00:28:26, metric 1
                      Receive
299920             *[LDP/9] 1w5d 01:26:08, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Pop
299920(S=0)        *[LDP/9] 1w5d 01:26:08, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Pop
299936             *[LDP/9] 1w5d 01:26:08, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Swap 300016
300096             *[LDP/9] 16:22:35, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Swap 300128
300112             *[LDP/9] 16:22:35, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Swap 300144
300128             *[LDP/9] 16:22:35, metric 1
                    > to 203.0.113.2 via ge-3/1/0.0, Swap 300160
300144             *[L2VPN/7] 16:22:33
                    > via ge-3/1/2.0, Pop       Offset: 4
ge-3/1/2.0         *[L2VPN/7] 16:22:33, metric2 1
                    > to 203.0.113.2 via ge-3/1/0.0, Push 300176, Push 300016(top) Offset: 252
```

From operational mode, run the `show route table ms-pw.l2vpn.1` command.

```
user@T-PE1> show route table ms-pw.l2vpn.1
ms-pw.l2vpn.1: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.10.10.10:15:700:0.0.2.188:700/160 AD2
                   *[BGP/170] 16:23:27, localpref 100, from 10.255.2.1
                      AS path: 2 I, validation-state: unverified
                    > to 203.0.113.2 via ge-3/1/0.0, Push 300016
10.10.10.10:15:800:0.0.3.32:800/160 AD2
                   *[L2VPN/170] 1w5d 23:25:19, metric2 1
                      Indirect
10.255.2.1:CtrlWord:5:100:15:700:0.0.2.188:700:800:0.0.3.32:800/304 PW2
                   *[LDP/9] 16:23:25
                      Discard
10.255.2.1:CtrlWord:5:100:15:800:0.0.3.32:800:700:0.0.2.188:700/304 PW2
                   *[L2VPN/7] 16:23:27, metric2 1
                    > to 203.0.113.2 via ge-3/1/0.0, Push 300016
```

Meaning

The output shows all the learned routes, including the autodiscovery (AD) routes.

The AD2 prefix format is `RD:SAII-type2`, where:

- `RD` is the route distinguisher value.
- `SAII-type2` is the type 2 source attachment identifier value.

The PW2 prefix format is `Neighbor_Addr:C:PWtype:l2vpn-id:SAII-type2:TAII-type2`, where:

- `Neighbor_Addr` is the loopback address of neighboring S-PE device.
- `C` indicates if Control Word (CW) is enabled or not.
  - `C` is `CtrlWord` if CW is set.
  - `C` is `NoCtrlWord` if CW is not set.
- `PWtype` indicates the type of the pseudowire.
  - `PWtype` is `4` if it is in Ethernet tagged mode.
  - `PWtype` is `5` if it is Ethernet only.

- `l2vpn-id` is the Layer 2 VPN ID for the MS-PW routing instance.
- `SAII-type2` is the type 2 source attachment identifier value.
- `TAII-type2` is the type 2 target attachment identifier value.

Verifying the LDP Database

Purpose

Verify the MS-PW labels received by T-PE1 from S-PE1 and sent from T-PE1 to S-PE1.

Action

From operational mode, run the `show ldp database` command.

```
user@T-PE1> show ldp database
Input label database, 10.255.10.1:0--10.255.2.1:0
  Label     Prefix
      3      10.255.2.1/32
 300112      10.255.3.1/32
 300128      10.255.4.1/32
 299968      10.255.10.1/32
 299904      10.255.13.1/32
 300144      10.255.14.1/32
 300176      FEC129 CtrlWord ETHERNET 000a0064:0000000f 000002bc:000002bc:000002bc
00000320:00000320:00000320

Output label database, 10.255.10.1:0--10.255.2.1:0
  Label     Prefix
 299936      10.255.2.1/32
 300096      10.255.3.1/32
 300112      10.255.4.1/32
      3      10.255.10.1/32
 299920      10.255.13.1/32
 300128      10.255.14.1/32
 300144      FEC129 CtrlWord ETHERNET 000a0064:0000000f 00000320:00000320:00000320
000002bc:000002bc:000002bc

Input label database, 10.255.10.1:0--10.255.13.1:0
  Label     Prefix
 300016      10.255.2.1/32
 300128      10.255.3.1/32
 300144      10.255.4.1/32
 300080      10.255.10.1/32
      3      10.255.13.1/32
 300160      10.255.14.1/32

Output label database, 10.255.10.1:0--10.255.13.1:0
  Label     Prefix
 299936      10.255.2.1/32
 300096      10.255.3.1/32
 300112      10.255.4.1/32
      3      10.255.10.1/32
 299920      10.255.13.1/32
 300128      10.255.14.1/32
```

Meaning

The labels with `FEC129` prefix are related to the MS-PW.

Checking the MS-PW Connections on T-PE1

Purpose

Make sure that all of the FEC 129 MS-PW connections come up correctly.

Action

From operational mode, run the `show l2vpn connections extensive` command.

```
user@T-PE1> show l2vpn connections extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
```

```
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: ms-pw
  L2vpn-id: 100:15
    Number of local interfaces: 1
    Number of local interfaces up: 1
    ge-3/1/2.0
  Local source-attachment-id: 800:0.0.3.32:800 (CE1)
    Target-attachment-id      Type  St     Time last up          # Up trans
    700:0.0.2.188:700         rmt   Up     Sep 18 01:10:55 2013           1
      Remote PE: 10.255.2.1, Negotiated control-word: Yes (Null)
      Incoming label: 300048, Outgoing label: 300016
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Local interface: ge-3/1/2.0, Status: Up, Encapsulation: ETHERNET
      Pseudowire Switching Points :
        Local address           Remote address        Status
        10.255.2.1              10.255.3.1            forwarding
        10.255.3.1              10.255.14.1           forwarding
    Connection History:
        Sep 18 01:10:55 2013  status update timer
        Sep 18 01:10:55 2013  PE route changed
        Sep 18 01:10:55 2013  Out lbl Update                    300016
        Sep 18 01:10:55 2013  In lbl Update                     300048
        Sep 18 01:10:55 2013  loc intf up                   ge-3/1/2.0
```

Check the following fields in the output to verify that MS-PW is established between the T-PE devices:

- **`Target-attachment-id`**—Check if the TAI value is the SAI value of T-PE2.

- **`Remote PE`**—Check if the T-PE2 loopback address is listed.
- **`Negotiated PW status TLV`**—Ensure that the value is `Yes`.
- **`Pseudowire Switching Points`**—Check if the switching points are listed from S-PE1 to S-PE2 and from S-PE2 to T-PE2.

Meaning

MS-PW is established between T-PE1 and T-PE2 in the forwarding direction.

Checking the MS-PW Connections on S-PE1

Purpose

Make sure that all of the FEC 129 MS-PW connections come up correctly for the mspw routing instance.

Action

From operational mode, run the `show l2vpn connections instance __MSPW__ extensive` command.

```
user@S-PE1> show l2vpn connections instance __MSPW__ extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
```

```
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: __MSPW__
  L2vpn-id: 100:15
  Local source-attachment-id: 700:0.0.2.188:700
    Target-attachment-id      Type  St     Time last up          # Up trans
    800:0.0.3.32:800          rmt   Up     Sep 18 01:17:38 2013           1
      Remote PE: 10.255.10.1, Negotiated control-word: Yes (Null), Encapsulation: ETHERNET
      Incoming label: 300016, Outgoing label: 300048
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
  Local source-attachment-id: 800:0.0.3.32:800
    Target-attachment-id      Type  St     Time last up          # Up trans
    700:0.0.2.188:700         rmt   Up     Sep 18 01:17:38 2013           1
      Remote PE: 10.255.3.1, Negotiated control-word: Yes (Null), Encapsulation: ETHERNET
      Incoming label: 300000, Outgoing label: 300064
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Pseudowire Switching Points :
        Local address          Remote address        Status
        10.255.3.1             10.255.14.1           forwarding
```

Check the following fields in the output to verify that MS-PW is established between the T-PE devices:

- **`Target-attachment-id`**—Check if the TAI value is the SAI value of T-PE2.
- **`Remote PE`**—Check if the T-PE1 and S-PE2 loopback addresses are listed.
- **`Negotiated PW status TLV`**—Ensure that the value is `Yes`.
- **`Pseudowire Switching Points`**—Check if the switching points are listed from S-PE2 to T-PE2.

Meaning

MS-PW is established between T-PE1 and T-PE2 in the forwarding direction.

Checking the MS-PW Connections on S-PE2

Purpose

Make sure that all of the FEC 129 MS-PW connections come up correctly for the mspw routing instance.

Action

From operational mode, run the `show l2vpn connections instance __MSPW__ extensive` command.

```
user@S-PE2> show l2vpn connections instance __MSPW__ extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: __MSPW__
  L2vpn-id: 100:15
  Local source-attachment-id: 700:0.0.2.188:700
    Target-attachment-id      Type  St     Time last up          # Up trans
    800:0.0.3.32:800          rmt   Up     Sep 18 00:58:55 2013          1
      Remote PE: 10.255.2.1, Negotiated control-word: Yes (Null), Encapsulation: ETHERNET
      Incoming label: 300064, Outgoing label: 300000
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Pseudowire Switching Points :
        Local address          Remote address        Status
        10.255.2.1             10.255.10.1           forwarding
  Local source-attachment-id: 800:0.0.3.32:800
    Target-attachment-id      Type  St     Time last up          # Up trans
    700:0.0.2.188:700         rmt   Up     Sep 18 00:58:55 2013          1
      Remote PE: 10.255.14.1, Negotiated control-word: Yes (Null), Encapsulation: ETHERNET
      Incoming label: 300048, Outgoing label: 300112
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
```

Check the following fields in the output to verify that MS-PW is established between the T-PE devices:

- `Target-attachment-id`—Check if the TAI value is the SAI value of T-PE1.
- `Remote PE`—Check if the S-PE1 and T-PE2 loopback addresses are listed.
- `Negotiated PW status TLV`—Ensure that the value is `Yes`.
- `Pseudowire Switching Points`—Check if the switching points are listed from S-PE1 to T-PE1.

Meaning

MS-PW is established between T-PE1 and T-PE2 in the reverse direction.

Checking the MS-PW Connections on T-PE2

Purpose

Make sure that all of the FEC 129 MS-PW connections come up correctly.

Action

From operational mode, run the `show l2vpn connections extensive` command.

```
user@T-PE2> show l2vpn connections extensive
Layer-2 VPN connections:
```

```
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: ms-pw
  L2vpn-id: 100:15
    Number of local interfaces: 1
    Number of local interfaces up: 1
    ge-2/0/0.0
  Local source-attachment-id: 700:0.0.2.188:700 (CE2)
    Target-attachment-id      Type  St     Time last up          # Up trans
    800:0.0.3.32:800          rmt   Up     Sep 18 01:35:21 2013           1
      Remote PE: 10.255.3.1, Negotiated control-word: Yes (Null)
      Incoming label: 300112, Outgoing label: 300048
      Negotiated PW status TLV: Yes
      local PW status code: 0x00000000, Neighbor PW status code: 0x00000000
      Local interface: ge-2/0/0.0, Status: Up, Encapsulation: ETHERNET
      Pseudowire Switching Points :
        Local address          Remote address        Status
        10.255.3.1             10.255.2.1            forwarding
        10.255.2.1             10.255.10.1           forwarding
    Connection History:
        Sep 18 01:35:21 2013  status update timer
```

```
          Sep 18 01:35:21 2013  PE route changed
          Sep 18 01:35:21 2013  Out lbl Update                    300048
          Sep 18 01:35:21 2013  In lbl Update                     300112
          Sep 18 01:35:21 2013  loc intf up                   ge-2/0/0.0
```

Check the following fields in the output to verify that MS-PW is established between the T-PE devices:

- `Target-attachment-id`—Check if the TAI value is the SAI value of T-PE1.
- `Remote PE`—Check if the T-PE1 loopback address is listed.
- `Negotiated PW status TLV`—Ensure that the value is `Yes`.
- `Pseudowire Switching Points`—Check if the switching points are listed from S-PE2 to S-PE1 and from S-PE1 to T-PE1.

Meaning

MS-PW is established between T-PE1 and T-PE2 in the reverse direction.

Troubleshooting

IN THIS SECTION

- Ping | 595
- Bidirectional Forwarding Detection | 596
- Traceroute | 597

To troubleshoot the MS-PW connection, see:

Ping

Problem

How to check the connectivity between the T-PE devices and between a T-PE device and an intermediary device.

Solution

Verify that T-PE1 can ping T-PE2. The **ping mpls l2vpn fec129** command accepts SAIs and TAIs as integers or IP addresses and also allows you to use the CE-facing interface instead of the other parameters (**instance**, **local-id**, **remote-id**, **remote-pe-address**).

Checking Connectivity Between T-PE1 and T-PE2

```
user@T-PE1> ping mpls l2vpn fec129 instance FEC129-VPWS local-id 800:800:800 remote-pe-address
10.255.14.1 remote-id 700:700:700
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
user@T-PE1> ping mpls l2vpn fec129 interface ge-3/1/2
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Checking Connectivity Between T-PE1 and S-PE2

```
user@T-PE1> ping mpls l2vpn fec129 interface ge-3/1/2 bottom-label-ttl 2
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Bidirectional Forwarding Detection

Problem

How to use BFD to troubleshoot the MS-PW connection from the T-PE device.

Solution

From operational mode, verify the show bfd session extensive command output.

```
user@T-PE1> show bfd session extensive

                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
198.51.100.7                Up        ge-3/1/0.0     0.900     0.300        3
```

```
  Client FEC129-OAM, TX interval 0.300, RX interval 0.300
  Session up time 03:12:42
  Local diagnostic None, remote diagnostic None
  Remote state Up, version 1
  Replicated
  Session type: VCCV BFD
  Min async interval 0.300, min slow interval 1.000
  Adaptive async TX interval 0.300, RX interval 0.300
  Local min TX interval 0.300, minimum RX interval 0.300, multiplier 3
  Remote min TX interval 0.300, min RX interval 0.300, multiplier 3
  Local discriminator 19, remote discriminator 19
  Echo mode disabled/inactive
  Remote is control-plane independent
  L2vpn-id 100:15, Local-id 800:0.0.3.32:800, Remote-id 700:0.0.2.188:700
   Session ID: 0x103

1 sessions, 1 clients
Cumulative transmit rate 3.3 pps, cumulative receive rate 3.3 pps
```

Traceroute

Problem

How to verify that MS-PW was established.

Solution

From operational mode, verify `traceroute` output.

```
user@T-PE1> traceroute mpls l2vpn fec129 interface interface
Probe options: ttl 64, retries 3, exp 7

  ttl    Label  Protocol    Address          Previous Hop     Probe Status
    1           FEC129      10.255.10.1         (null)           Success

    2           FEC129      10.255.2.1       10.255.10.1         Success

    3           FEC129      10.255.3.1       10.255.2.1          Success

    4           FEC129      10.255.14.1      10.255.2.1          Egress
```

```
Path 1 via ge-3/1/2 destination 198.51.100.0
```

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic

This topic shows how to configure flow-aware transport of pseudowires (FAT) flow labels for forwarding equivalence class (FEC) 128 virtual private wire service (VPWS) pseudowires for load-balancing MPLS traffic.

Before you begin:

1. Configure the device interfaces and enable MPLS on all the interfaces.
2. Configure MPLS and an LSP to the remote PE router.
3. Configure OSPF and IS-IS.
4. Configure LDP on the loopback interface and the PE interface connecting to the P (transit) router.

FAT flow labels enable load-balancing of MPLS packets across equal-cost multipath (ECMP) paths or link aggregation groups (LAGs) without the need for deep packet inspection of the payload. FAT flow labels can be used for LDP-signaled FEC 128 and FEC 129 pseudowires for virtual private LAN service (VPLS) and VPWS networks.

You can configure FAT flow labels to be signaled by LDP on FEC 128 VPWS pseudowires by including the `flow-label-transmit` and `flow-label-receive` configuration statements at the `[edit protocols l2circuit neighbor` *`neighbor-id`* `interface` *`interface-name`*`]` hierarchy level. This configuration sets the T bit and R bit advertisement to 1 (the default being 0) in the Sub-TLV field, which is one of the interface parameters of the FEC for the LDP label-mapping message header. These statements signal the pushing and popping of the load-balancing label to the routing peers in the control plane.

Alternatively, you can configure the following statements at the `[edit protocols l2circuit neighbor` *`neighbor-id`* `interface` *`interface-name`*`]` hierarchy level:

- `flow-label-transmit-static` to statically push the flow label on the pseudowire packets sent to the remote provider edge (PE) router.
- `flow-label-receive-static` to statically pop the flow label on the pseudowire packets received from the remote PE router.

To configure the FAT flow label for an FEC 128-signaled VPLS pseudowire, on the ingress PE router:

1. Configure the neighbors for the Layer 2 circuit.

   ```
   [edit protocols l2circuit]
   user@PE1# set neighbor neighbor-id
   ```

   All the Layer 2 circuits using a particular remote PE router designated for remote CE routers are listed under the `neighbor` statement. Each neighbor is identified by its IP address and is usually the end-point destination for the label-switched path (LSP) tunnel transporting the Layer 2 circuit.

2. Configure the interface for the Layer 2 circuit neighbor and a unique identifier for the Layer 2 circuit.

   ```
   [edit protocols l2circuit neighbor neighbor-id]
   user@PE1# set interface interface-name virtual-circuit-id unique-l2ckt-identifier
   ```

3. **NOTE**: You can only configure one of the following pairs of statements:

   - `flow-label-transmit` and `flow-label-receive` **or**,
   - `flow-label-transmit-static` and `flow-label-receive-static`

   Configure the router to signal the capability to push the flow label in the transmit direction to the remote PE router.

   ```
   [edit protocols l2circuit neighbor neighbor-id interface interface-name]
   user@PE1# set flow-label-transmit
   ```

4. Alternatively, configure the `flow-label-transmit-static` statement to statically push the flow label on the pseudowire packets sent to the remote PE router.

   ```
   [edit protocols l2circuit neighbor neighbor-id interface interface-name]
   user@PE1# set flow-label-transmit-static
   ```

   If the incoming pseudowire packet is not marked with the flow label, the packet is dropped by the egress PE router.

5. Configure the router to signal the capability to pop the flow label in the receive direction to the remote egress PE router.

   ```
   [edit protocols l2circuit neighbor neighbor-id interface interface-name]
   user@PE1# set flow-label-receive
   ```

6. Alternatively, configure the `flow-label-receive-static` statement to pop the flow label on the pseudowire packets received from the remote PE router.

   ```
   [edit protocols l2circuit neighbor neighbor-id interface interface-name]
   user@PE1# set flow-label-receive-static
   ```

   The ingress PE router inserts the flow label in the pseudowire packet, irrespective of the information exchanged in the signaling plane. If the egress PE router cannot handle the pseudowire packet marked with the flow label, the packet is dropped.

7. Verify and commit the configuration.

   For example:

   ```
   [edit protocols l2circuit]
   user@PE1# show
   neighbor 10.255.104.135 {
       interface ge-1/0/8.0 {
           virtual-circuit-id 1;
           flow-label-transmit;
           flow-label-receive;
       }
   }
   ```

   OR:

   ```
   [edit protocols l2circuit]
   user@PE1# show
   neighbor 10.255.104.135 {
       interface ge-1/0/8.0 {
           virtual-circuit-id 1;
           flow-label-transmit-static;
           flow-label-receive-static;
       }
   }
   ```

8. Repeat the configuration on the remote egress PE router.

RELATED DOCUMENTATION

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | **770**

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | **601**

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | **772**

FAT Flow Labels Overview | **496**

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic

This topic shows how to configure flow-aware transport of pseudowires (FAT) flow labels for forwarding equivalence class (FEC) 129 virtual private wire service (VPWS) pseudowires.

Before you begin:

1. Configure the device interfaces and enable MPLS on all core-facing interfaces.
2. Configure CCC encapsulation and the CCC address family for interfaces configured as members of the FEC 129 VPWS instance.
3. Configure MPLS and an LSP to the remote provider edge (PE) router.
4. Configure the BGP sessions on the PE devices with the BGP autodiscovery-only address family to allow exchange of the autodiscovery routes.
5. Configure an IGP such as IS-IS or OSPF.
6. Configure LDP on the loopback interface and the core-facing interface.
7. Configure the autonomous system (AS) number.

FAT flow labels enable load-balancing of MPLS packets across equal-cost multipath (ECMP) paths or link aggregation groups (LAGs) without the need for deep packet inspection of the payload. FAT flow labels can be used for LDP-signaled FEC 128 and FEC 129 pseudowires for virtual private LAN service (VPLS) and VPWS networks.

You can configure FAT flow labels to be signaled by LDP on FEC 129 VPWS pseudowires (Layer 2 circuits) by including the `flow-label-transmit` and `flow-label-receive` configuration statements at the `[edit routing-instances *instance-name* protocols l2vpn site *name*]` or the `[edit routing-instances *instance-name* protocols l2vpn site *name* interface *interface-name*]` hierarchy level. This configuration sets the T bit and R bit advertisement to 1 (the default being 0) in the Sub-TLV field, which is one of the interface parameters of the FEC for the LDP label-mapping message header. These statements signal the pushing and popping of the load-balancing label to the routing peers in the control plane.

To configure the FAT flow label for an FEC 129 VPWS pseudowire, on the ingress PE router:

1. Configure the VPWS routing instance.

   LDP listens for routes from instance.l2vpn.0 for any instance configured for FEC 129 VPWS. These routes are identified by the `instance-type l2vpn` statement in the routing instance and the presence of the `l2vpn-id` statement.

   ```
   [edit ]
   user@PE1# set routing-instances instance-name instance-type l2vpn
   user@PE1# set routing-instances instance-name interface interface-name
   user@PE1# set routing-instances instance-name route-distinguisher (as-number:x:y | ip-
   address:id)
   user@PE1# set routing-instances instance-name l2vpn-id (as-number:x:y | ip-address:id)
   user@PE1# set routing-instances instance-name vrf-target community
   user@PE1# set routing-instances instance-name protocols l2vpn site site-name source-
   attachment-identifier identifier
   user@PE1# set routing-instances instance-name protocols l2vpn site site-name interface
   interface-name target-attachment-identifier identifier
   ```

   Because VPWS is a point-to-point service, FEC 129 VPWS routing instances are configured as `instance-type l2vpn`. As with FEC 129 VPLS, FEC 129 VPWS uses the `l2vpn-id` statement to define the Layer 2 VPN of which the routing instance is a member. The presence of the `l2vpn-id` statement designates that FEC 129 LDP-signaling is used for the routing instance.

2. Configure the device to signal the capability to push the flow label in the transmit direction to the remote PE router.

   ```
   [edit routing-instances instance-name protocols l2vpn site name interface interface-name]
   user@PE1# set flow-label-transmit
   ```

3. Configure the device to signal the capability to pop the flow label in the receive direction to the remote PE router.

```
[edit routing-instances instance-name protocols l2vpn site name interface interface-name]
user@PE1# set flow-label-receive
```

4. Alternatively, configure the `flow-label-transmit` and `flow-label-receive` statements directly within the site. When configured within the site, the defined parameters affect any pseudowire originating from that site. When configured under an interface within the site, the defined parameters affect that single specific pseudowire. This enables you to manipulate the parameters across all pseudowires associated with a particular local site in one place in the configuration.

```
[edit routing-instances instance-name protocols l2vpn site name ]
user@PE1# set flow-label-receive
user@PE1# set flow-label-transmit
```

5. Verify and commit the configuration.

   For example:

```
[edit routing-instances FEC129-VPWS]
user@PE1# show
instance-type l2vpn;
interface ge-0/0/1.600;
route-distinguisher 10.255.255.1:100;
l2vpn-id l2vpn-id:100:100;
vrf-target target:100:100;
protocols {

    l2vpn {
        site ONE {
            source-attachment-identifier 1;
            interface ge-0/0/1.600 {
                target-attachment-identifier 2;
                flow-label-transmit; <<< Applicable only to the pseudowire specific to the
interface >>>
                flow-label-receive;
            }
        }
        site TWO {
            source-attachment-identifier 3;
            flow-label-transmit; <<< Applicable to all pseudowires within the site >>>
            flow-label-receive;
            interface ge-0/0/2.600 {
                target-attachment-identifier 1;
            }
            interface ge-0/0/2.601 {
                target-attachment-identifier 4;
            }
        }
    }
}
```

6. Repeat the configuration on the remote egress PE router.

RELATED DOCUMENTATION

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | **598**

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | **770**

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | **772**

FAT Flow Labels Overview | **496**

7

PART

Configuring VPLS

- Overview | 606
- VPLS Configuration Overview | 609
- Configuring Signaling Protocols for VPLS | 622
- Assigning Routing Instances to VPLS | 675
- Associating Interfaces with VPLS | 734
- Configuring Pseudowires | 751
- Configuring Multihoming | 885
- Configuring Point-to-Multipoint LSPs | 1028
- Configuring Inter-AS VPLS and IRB VPLS | 1101
- Configuring Load Balancing and Performance | 1145
- Configuring Class of Service and Firewall Filters in VPLS | 1237
- Monitoring and Tracing VPLS | 1263

CHAPTER 30

Overview

IN THIS CHAPTER

- Introduction to VPLS | **606**
- Supported VPLS Standards | **607**
- Supported Platforms and PICs | **607**

Introduction to VPLS

VPLS is an Ethernet-based point-to-multipoint Layer 2 VPN. It allows you to connect geographically dispersed Ethernet local area networks (LAN) sites to each other across an MPLS backbone. For customers who implement VPLS, all sites appear to be in the same Ethernet LAN even though traffic travels across the service provider's network.

VPLS, in its implementation and configuration, has much in common with a Layer 2 VPN. In VPLS, a packet originating within a service provider customer’s network is sent first to a customer edge (CE) device (for example, a router or Ethernet switch). It is then sent to a provider edge (PE) router within the service provider’s network. The packet traverses the service provider’s network over a MPLS label-switched path (LSP). It arrives at the egress PE router, which then forwards the traffic to the CE device at the destination customer site.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The difference is that for VPLS, packets can traverse the service provider’s network in point-to-multipoint fashion, meaning that a packet originating from a CE device can be broadcast to all the PE routers participating in a VPLS routing instance. In contrast, a Layer 2 VPN forwards packets in point-to-point fashion only.

The paths carrying VPLS traffic between each PE router participating in a routing instance are called pseudowires. The pseudowires are signaled using either BGP or LDP.

Supported VPLS Standards

Junos OS substantially supports the following Internet RFCs and draft, which define standards for virtual private LAN service (VPLS).

- RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*
- RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*

  FEC 128, FEC 129, control bit `0`, the Ethernet pseudowire type `0x0005`, and the Ethernet tagged mode pseudowire type `0x0004` are supported.
- RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*
- RFC 6790, *The Use of Entropy Labels in MPLS Forwarding*
- Internet draft draft-kompella-l2vpn-vpls-multihoming, *Multi-homing in BGP-based Virtual Private LAN Service*

RELATED DOCUMENTATION

Supported Carrier-of-Carriers and Interprovider VPN Standards
Supported VPWS Standards | **495**
Supported Layer 2 VPN Standards | **146**
Supported Layer 3 VPN Standards
Supported Multicast VPN Standards
Accessing Standards Documents on the Internet

Supported Platforms and PICs

Virtual private LAN service (VPLS) is supported on all M Series routers except the M160.

VPLS is supported on all MX Series and T Series routers.

VPLS is supported on the following SRX Services Gateways for the branch:

- SRX100
- SRX210
- SRX240

- SRX650

VPLS is supported on the following PICs:

- All ATM2 IQ PICs
- 4-port Fast Ethernet PIC with 10/100 Base-TX interfaces PIC
- 1-port, 2-port, and 10-port Gigabit Ethernet PICs
- 1-port, 2-port, and 4-port Gigabit Ethernet PICs with SFP
- 1-port 10-Gigabit Ethernet PIC
- 1-port and 2-port Gigabit Ethernet Intelligent Queuing (IQ) PICs
- 4-port and 8-port Gigabit Ethernet IQ2 PICs with SFP
- 1-port 10-Gigabit Ethernet IQ2 PIC with XFP
- 4-port, quad-wide Gigabit Ethernet PIC
- 10-port 10-Gigabit OSE PIC

CHAPTER 31

VPLS Configuration Overview

IN THIS CHAPTER

- Introduction to Configuring VPLS | **609**
- Configuring an Ethernet Switch as the CE Device for VPLS | **621**

Introduction to Configuring VPLS

IN THIS SECTION

- Understanding VPLS Interfaces | **610**
- VPLS CLI Configuration Hierarchy Differences between Junos OS and Junos OS Evolved | **611**
- Sample VPLS Configuration on Junos OS | **615**
- Sample VPLS Configuration on Junos OS Evolved | **618**
- Platform-Specific VPLS Behavior | **620**

Virtual private LAN service (VPLS) allows you to provide a point-to-multipoint LAN between a set of sites in a virtual private network (VPN).

To configure VPLS functionality, you must enable VPLS support on the provider edge (PE) router. You must also configure PE routers to distribute routing information to the other PE routers in the VPLS. In addition, you must also configure the circuits between the PE routers and the customer edge (CE) routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

You can support VPLS traffic by configuring a routing instance with the instance-type of `vpls` on Junos OS or a routing instance on with the instance-type of `virtual-switch`Junos Evolve OS. . The routing instance carries Ethernet traffic transparently across the service provider’s network. As with other routing instances, all logical interfaces belonging to a VPLS routing instance are listed under that instance.

NOTE: When you configure normalized VLANs under a **virtual-switch** routing instance, either by not configuring vlan IDs or including the `vlan-id none` statement, then you must also include the `service-type single` statement at the `[edit routing-instances` *`routing-instance`* `protocol vpls]` hierarchy.

In addition to VPLS routing instance configuration, you must configure MPLS label-switched paths (LSPs) between the PE routers, IBGP sessions between the PE routers, and an interior gateway protocol (IGP) on the PE and provider (P) routers.

Many configuration procedures for VPLS are identical to the procedures for Layer 2 VPNs and Layer 3 VPNs.

Understanding VPLS Interfaces

Interface names:

For each VPLS routing instance on a PE router, you specify which interfaces are to be used to carry VPLS traffic between the PE and CE devices.

Specify both the physical and logical portions of the interface name, in the following format: `physical.logical`. For example, in ge-1/0/1.2, ge-1/0/1 is the physical portion of the interface name and 2 is the logical portion. If you do not specify the logical portion of the interface name, 0 is set by default.

A logical interface can be associated with only one routing instance.

Encapsulation Type:

The link-layer encapsulation type for a VPLS interface can be one of the following:

- `ethernet-vpls`: Use Ethernet VPLS encapsulation on Ethernet interfaces that have VPLS enabled and that must accept packets carrying standard Tag Protocol Identifier (TPID) values.

  `extended-vlan-vpls`: Use extended virtual LAN (VLAN) VPLS encapsulation on Ethernet interfaces that have VLAN 802.1Q tagging and VPLS enabled and that must accept packets carrying TPIDs 0x8100, 0x9100, and 0x9901. All VLAN IDs from 1 through 1023 are valid for VPLS VLANs on Fast Ethernet interfaces, and all VLAN IDs from 1 through 4094 are valid for VPLS VLANs on Gigabit Ethernet interfaces.

`vlan-vpls`: Use VLAN VPLS encapsulation on Ethernet interfaces with VLAN tagging and VPLS enabled. Interfaces with VLAN VPLS encapsulation accept packets carrying standard TPID values only. You must configure this encapsulation type on both the physical interface and the logical interface. VLAN IDs 1 through 511 are reserved for normal Ethernet VLANs, IDs 512 through 1023 are reserved for VPLS VLANs on Fast Ethernet interfaces, and IDs 512 through 4094 are reserved for VPLS VLANs on Gigabit Ethernet interfaces.

- `flexible-ethernet-services`: Use flexible Ethernet services encapsulation when you want to configure multiple per-unit Ethernet encapsulations. This encapsulation type allows you to configure any combination of route, TCC, CCC, and VPLS encapsulations on a single physical port. Aggregated Ethernet bundles cannot use this encapsulation type.

  For flexible Ethernet services encapsulation, VLAN IDs from 1 through 511 are no longer reserved for normal VLANs.

VPLS CLI Configuration Hierarchy Differences between Junos OS and Junos OS Evolved

When you configure VPLS on devices running on Junos OS or Junos OS Evolved, be aware of the CLI hierarchy differences. Table 13 on page 612 explains the VPLS CLI hierarchy configuration differences on devices running on Junos OS and Junos OS Evolved.

To know about the devices running on Junos OS, see https://www.juniper.net/documentation/product/us/en/junos-os/#cat=by_product.

To know about the devices running on Junos OS Evolved, see https://www.juniper.net/documentation/product/us/en/junos-os-evolved/#cat=hardware_documentation.

Table 13: VPLS Configuration CLI Differences between Junos OS and Junos OS Evolved

Task	Junos OS	Junos OS Evolved

Enable VPLS routing instance	`user@host# set  routing-instances routing-instance-name instance-type vpls`
	`user@host# set routing-instances routing-instance-name protocols vpls neighbor IP-address`
	`user@host# set routing-instances routing-instance-name protocols vpls no-tunnel-services`
	`user@host# set routing-instances routing-instance-name protocols vpls vpls-id number`
	`user@host# set routing-instances routing-instance-name vlan-id number`
	`user@host# set routing-instances routing-instance-name interface interface-name`
		`user@host# set  routing-instances routing-instance-name instance-type virtual-switch`
		`user@host# set routing-instances routing-instance-name protocols vpls neighbor IP-address`
		`user@host# set routing-instances routing-instance-name protocols vpls no-tunnel-services`
		`user@host# set routing-instances routing-instance-name protocols vpls vpls-id number`
		`user@host# set routing-instances routing-instance-name vlans vlan-name vlan-id number`
		`user@host# set routing-instances routing-instance-name vlans vlan-name interface interface-name`
		
		NOTE: You need to mandatorily configure the `no-tunnel-services` configuration because tunnel interface is not supported on Junos OS Evolved.

Table 13: VPLS Configuration CLI Differences between Junos OS and Junos OS Evolved *(Continued)*

Task	Junos OS	Junos OS Evolved

Configure VLAN without VLAN IDs, or with VLAN ID as `none`, or VLAN ID as `all`	`user@host# set routing-instances routing-instance-name`	• Configure VLAN without VLAN IDs
		`user@host# set routing-instances routing-instance-name vlans vlan-name`
		NOTE: Support for multiple bridge domain within a routing-instance varies across platforms. On platforms that do not support multiple bridge domains, you must include the `service-type single` configuration statement at the `[edit routing-instances routing-instance protocol vpls]` hierarchy.
		• Configure VLAN with VLAN ID as `none` or `all`
		`user@host# set routing-instances routing-instance-name vlans vlan-name vlan-id none`
		`user@host# set routing-instances routing-instance-name vlans vlan-name protocols vpls service-type single`
		NOTE: When you configure a VLAN without VLAN IDs, or with VLAN ID as `none`, or VLAN ID as `all`, you must include the `service-type single` configuration

Table 13: VPLS Configuration CLI Differences between Junos OS and Junos OS Evolved *(Continued)*

Task	Junos OS	Junos OS Evolved
		statement at the `[edit routing-instances routing-instance protocol vpls]` hierarchy.
Configure VPLS family on interfaces	`user@host# set interfaces interface-name unit number family vpls`	Devices running Junos OS Evolved do not support configuring the protocol family of `VPLS` on the interfaces.

Configure VPLS encapsulation on the interfaces	`user@host# set interfaces interface-name flexible-vlan-tagging`
	`user@host# set interfaces interface-name speed 100g`
	`user@host# set interfaces interface-name encapsulation flexible-ethernet-services`
	`user@host# set interfaces interface-name unit number encapsulation vlan-vpls`
	`user@host# set interfaces interface-name unit number vlan-id number`
		`user@host# set interfaces interface-name flexible-vlan-tagging`
		`user@host# set interfaces interface-name speed 100g`
		`user@host# set interfaces interface-name encapsulation flexible-ethernet-services`
		`user@host# set interfaces interface-name unit number encapsulation vlan-vpls`
		`user@host# set interfaces interface-name unit number vlan-id number`

Configure VPLS encapsulation on the interfaces using bridge domain	`user@host# set interfaces et-0/0/6 flexible-vlan-tagging`
	`user@host# set interfaces et-0/0/6 encapsulation flexible-ethernet-services`
	`user@host# set interfaces et-0/0/6 unit 0 encapsulation vlan-bridge`
	`user@host# set interfaces et-0/0/6 unit 0 vlan-id 1`
		`user@host# set interfaces et-0/0/6 flexible-vlan-tagging`
		`user@host# set interfaces et-0/0/6 encapsulation flexible-ethernet-services`
		`user@host# set interfaces et-0/0/6 unit 0 encapsulation vlan-bridge`
		`user@host# set interfaces et-0/0/6 unit 0 vlan-id 1`

Table 13: VPLS Configuration CLI Differences between Junos OS and Junos OS Evolved *(Continued)*

Task	Junos OS	Junos OS Evolved

Configure an IRB on VPLS	`user@host# set routing-instances` *`routing-instance-name`* `instance-type vpls`
	`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`ip-address`*
	`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls no-tunnel-services`
	`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls vpls-id` *`number`*
	`user@host# set routing-instances` *`routing-instance-name`* `vlan-id` *`number`*
	`user@host# set routing-instances` *`routing-instance-name`* `interface` *`interface-name`*
	`user@host# set routing-instances` *`routing-instance-name`* `routing-interface` *`interface-name`*
	`user@host# set routing-instances` *`routing-instance-name`* `route-distinguisher` *`distinguisher`*
	`user@host# set routing-instances` *`routing-instance-name`* `vrf-target` *`target;`*
		`user@host# set routing-instances` *`routing-instance-name`* `instance-type virtual-switch`
		`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`ip-address`*
		`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls no-tunnel-services`
		`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls control-word`
		`user@host# set routing-instances` *`routing-instance-name`* `protocols vpls vpls-id` *`number`*
		`user@host# set routing-instances` *`routing-instance-name`* `vlans` *`vlan-name`* `vlan-id` *`number`*
		`user@host# set routing-instances` *`routing-instance-name`* `vlans` *`vlan-name`* `interface` *`interface-name`*
		`user@host# set  routing-instances` *`routing-instance-name`* `l3-interface` *`interface-name`*

Sample VPLS Configuration on Junos OS

The following is a sample VPLS configuration on Junos OS:

Configuration

```
[edit]user@host# set interfaces xe-0/0/2 flexible-vlan-tagging
user@host# set interfaces xe-0/0/2 speed 40g
user@host# set interfaces xe-0/0/2 encapsulation flexible-ethernet-services
user@host# set interfaces xe-0/0/2 unit 108 encapsulation vlan-vpls
```

```
user@host# set interfaces xe-0/0/2  unit 108 vlan-id 108
user@host# set interfaces xe-0/0/2  unit 108 family vpls
user@host# set routing-instances VPLS instance-type vpls
user@host# set routing-instances VPLS interface xe-0/0/2.108
user@host# set routing-instances VPLS protocols vpls interface xe-0/0/2.108
user@host# set routing-instances VPLS protocols vpls neighbor 192.168.1.3
user@host# set routing-instances VPLS protocols vpls no-tunnel-services
user@host# set routing-instances VPLS protocols vpls vpls-id 1001
```

Result

```


user@host> show configuration interfaces xe-0/0/2
flexible-vlan-tagging;
speed 40g;
encapsulation flexible-ethernet-services;
}
unit 108 {
    encapsulation vlan-vpls;
    vlan-id 108;
    family vpls;
}

user@host> show configuration routing-instances
VPLS {
    protocols {
        vpls {
            interface xe-0/0/2.108;
            neighbor 192.168.1.3;
            no-tunnel-services;
            vpls-id 1001;
        }
    }
    instance-type vpls;
    interface xe-0/0/2.108;
}
```

Verification

Confirm that the configuration is working properly.

Verify the VPLS connection status by running the `show vpls connections` command.

```
user@host> show vpls connections Layer-2 VPN connections:
Legend for interface status
Up -- operational
Dn -- down

Instance: VPLS
  VPLS-id: 1001
    Neighbor                  Type  St     Time last up          # Up trans
    192.168.1.3(vpls-id 1001)     rmt   Up     Jun 12 12:50:07 2023           1
      Remote PE: 192.168.1.3, Negotiated control-word: No
      Incoming label: 259, Outgoing label: 22
      Negotiated PW status TLV: No
      Local interface: lsi.1049600, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls VPLS neighbor 192.168.1.3 vpls-id 1001
      Flow Label Transmit: No, Flow Label Receive: No
```

In this output, the VPLS connection status is up between the two PE routers.

Verify the MAC address learnt on the local PE and the remote PE by running the `show vpls mac-table` command.

```
user@host> show vpls mac-table
MAC flags       (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
    O -OVSDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned
MAC)

Routing instance : VPLS
Bridging domain : __VPLS__, VLAN : NA
   MAC                 MAC      Logical          NH     MAC         active
   address             flags    interface        Index  property    source
   00:00:00:42:98:47   D        lsi.1049346
   00:00:00:42:98:4a   D        xe-0/0/2.108
```

In this output, you can see the MAC address (00:00:00:42:98:47) is learned from the local PE interface (et-0/0/4) with VLAN ID 108 and the MAC address (00:00:00:42:98:4a) is learned from the remote PE interface (lsi.1048832) with VLAN ID 108.

Sample VPLS Configuration on Junos OS Evolved

The following is a sample VPLS configuration on Junos OS Evolved:

Configuration

```
[edit]
user@host# set interfaces et-0/0/4 flexible-vlan-tagging
user@host# set interfaces et-0/0/4 speed 40g
user@host# set interfaces et-0/0/4 encapsulation flexible-ethernet-services
user@host# set interfaces et-0/0/4 unit 108 encapsulation vlan-bridge
user@host# set interfaces et-0/0/4 unit 108 vlan-id 108
user@host# set routing-instances VPLS instance-type virtual-switch
user@host# set routing-instances VPLS protocols vpls neighbor 192.168.1.3
user@host# set routing-instances VPLS protocols vpls no-tunnel-services
user@host# set routing-instances VPLS protocols vpls vpls-id 1001
user@host# set routing-instances VPLS vlans v108 vlan-id 108
user@host# set routing-instances VPLS vlans v108 interface et-0/0/4.108
```

Result

```
user@host> show configuration interfaces et-0/0/4
flexible-vlan-tagging;
speed 40g;
encapsulation flexible-ethernet-services;
unit 108 {
    encapsulation vlan-bridge;
    vlan-id 108;
}

user@host> show configuration routing-instances
VPLS {
    instance-type virtual-switch;
    protocols {
        vpls {
            neighbor 192.168.1.3;
            no-tunnel-services;
            vpls-id 1001;
        }
    }
    vlans {
        V108 {
            vlan-id 108;
            interface et-0/0/4.108;
        }
    }
}
```

Verification

Confirm that the configuration is working properly.

Verify the VPLS connection status by running the `show vpls connections` command.

```
user@host> show vpls connections
Layer-2 VPN connections:

Legend for interface status
Up -- operational
Dn -- down

Instance: VPLS
  VPLS-id: 1001
    Neighbor                  Type  St     Time last up          # Up trans
    192.168.1.3(vpls-id 1001)     rmt   Up     Jun 12 12:50:07 2023           1
      Remote PE: 192.168.1.3, Negotiated control-word: No
      Incoming label: 22, Outgoing label: 259
      Negotiated PW status TLV: No
      Local interface: lsi.1048576, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls VPLS neighbor 192.168.1.3 vpls-id 1001
      Flow Label Transmit: No, Flow Label Receive: No
```

In this output, the VPLS connection status is up between the two PE routers.

Verify the MAC address learnt on the local PE and the remote PE by running the `show ethernet-switching table instance VPLS` command.

```
user@host> show ethernet-switching table instance VPLS

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static, C -
Control MAC
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC
           GBP - group based policy)


Ethernet switching table : 2 entries, 2 learned
Routing instance : VPLS
    Vlan                MAC                 MAC         Age    GBP     Logical
NH        RTR
    name                address             flags              Tag     interface
Index     ID
    V108                00:00:00:42:98:47   D            -              et-0/0/4.108
0         0
    V108                00:00:00:42:98:4a   D            -              lsi.1048832
0         0
```

In this output, you can see the MAC address (00:00:00:42:98:47) is learned from the local PE interface (et-0/0/4) with VLAN ID 108 and the MAC address (00:00:00:42:98:4a) is learned from the remote PE interface (lsi.1048832) with VLAN ID 108.

Platform-Specific VPLS Behavior

Use Feature Explorer to confirm platform and release support for VPLS

Table 14: Platform-Specific Behavior

Platform	Difference
ACX 7000 Series	The ACX7000 family of routers do not support the `vlan-id all` statement.
ACX 7000 Series	The ACX7000 family of routers support a single bridge domain within a single virtual switch routing instance.
PTX10002-36QDD	The PTX10002-36QDD router supports multiple bridge domains within a single virtual switch routing instance.

Configuring an Ethernet Switch as the CE Device for VPLS

For VPLS configurations, the CE device does not necessarily need to be a router. You can link the PE routers directly to Ethernet switches. However, there are a few configuration issues to be aware of:

- When you configure VPLS routing instances and establish two or more connections between a CE Ethernet switch and a PE router, you must enable the Spanning Tree Protocol (STP) on the switch to prevent loops.
- The Junos OS allows standard Bridge Protocol Data Unit (BPDU) frames to pass through emulated Layer 2 connections, such as those configured with Layer 2 VPNs, Layer 2 circuits, and VPLS instances. However, CE Ethernet switches that generate proprietary BPDU frames might not be able to run STP across Juniper Networks routing platforms configured for these emulated Layer 2 connections.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

CHAPTER 32

Configuring Signaling Protocols for VPLS

IN THIS CHAPTER

- VPLS Routing and Virtual Ports | **622**
- BGP Signaling for VPLS PE Routers Overview | **624**
- Control Word for BGP VPLS Overview | **625**
- Configuring a Control Word for BGP VPLS | **626**
- BGP Route Reflectors for VPLS | **628**
- Interoperability Between BGP Signaling and LDP Signaling in VPLS | **629**
- Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS | **632**
- Example: VPLS Configuration (BGP Signaling) | **637**
- Example: VPLS Configuration (BGP and LDP Interworking) | **653**

VPLS Routing and Virtual Ports

Because VPLS carries Ethernet traffic across a service provider network, it must mimic an Ethernet network in some ways. When a PE router configured with a VPLS routing instance receives a packet from a CE device, it first determines whether it has the destination of the VPLS packet in the appropriate routing table. If it does, it forwards the packet to the appropriate PE router or CE device. If it does not, it broadcasts the packet to all other PE routers and CE devices that are members of that VPLS routing instance. In both cases, the CE device receiving the packet must be different from the one sending the packet.

NOTE: In the VPLS documentation, the term *router* is used to refer to any device that provides routing functions.

When a PE router receives a packet from another PE router, it first determines whether it has the destination of the VPLS packet in the appropriate routing table. If it does, the PE router either forwards the packet or drops it depending on whether the destination is a local or remote CE device:

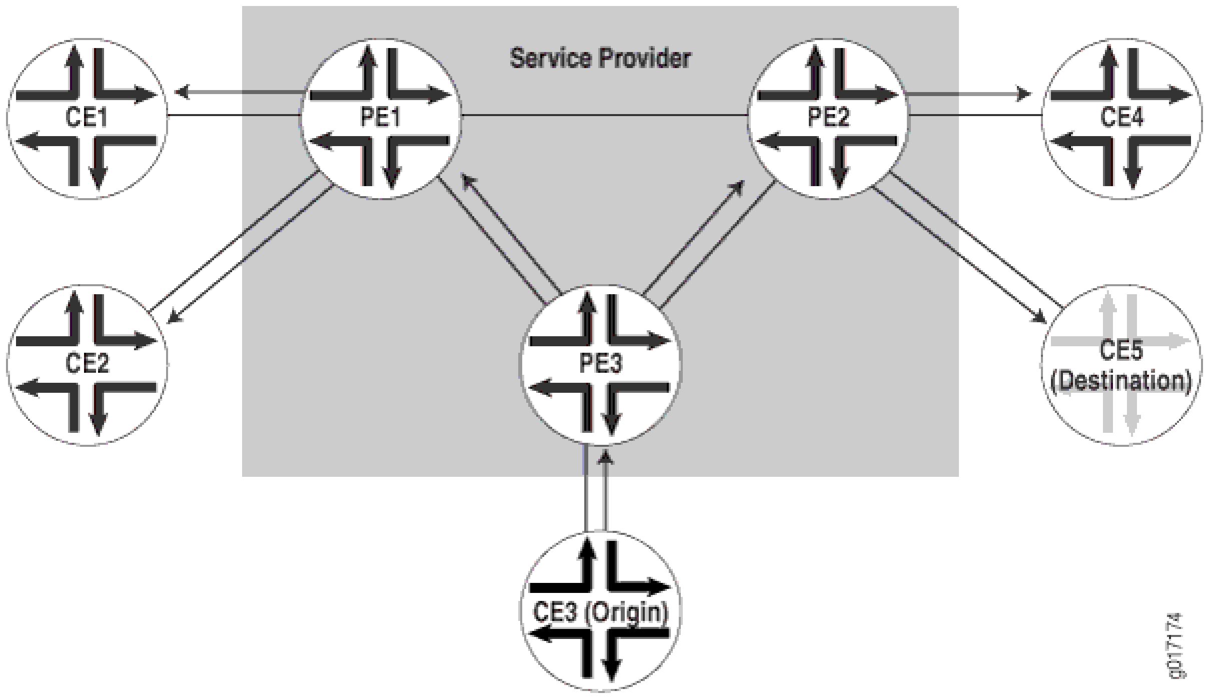
- If the destination is a local CE device, the PE router forwards the packet to it.

- If the destination is a remote CE device (connected to another PE router), the PE router discards the packet.

If the PE router cannot determine the destination of the VPLS packet, it floods the packet to all attached CE devices.

This process is illustrated in Figure 49 on page 623.

Figure 49: Flooding a Packet with an Unknown Destination to All PE Routers in the VPLS Instance



VPLS can be directly connected to an Ethernet switch. Layer 2 information gathered by an Ethernet switch (for example, media access control [MAC] addresses and interface ports) is included in the VPLS routing instance table. However, instead of all VPLS interfaces being physical switch ports, the router allows remote traffic for a VPLS instance to be delivered across an MPLS LSP and arrive on a virtual port. The virtual port emulates a local, physical port. Traffic can be learned, forwarded, or flooded to the virtual port in almost the same way as traffic is sent to a local port.

The VPLS routing table learns MAC address and interface information for both physical and virtual ports. The main difference between a physical port and a virtual port is that the router captures additional information from the virtual port, an outgoing MPLS label used to reach the remote site and an incoming MPLS label for VPLS traffic received from the remote site. The virtual port is generated dynamically on a Tunnel Services *Physical Interface Card* (PIC) when you configure VPLS on the router.

You can also configure VPLS without a Tunnel Services PIC. To do so, you use a label-switched interface (LSI) to provide VPLS functionality. An LSI MPLS label is used as the inner label for VPLS. This label maps to a VPLS routing instance. On the PE router, the LSI label is stripped and then mapped to a logical LSI interface. The Layer 2 Ethernet frame is then forwarded using the LSI interface to the correct VPLS routing instance.

One restriction on flooding behavior in VPLS is that traffic received from remote PE routers is never forwarded to other PE routers. This restriction helps prevent loops in the core network. However, if a CE Ethernet switch has two or more connections to the same PE router, you must enable the Spanning Tree Protocol (STP) on the CE switch to prevent loops. STP is supported on MX Series routers and EX Series switches only.

The Junos OS allows standard Bridge Protocol Data Unit (BPDU) frames to pass through emulated Layer 2 connections, such as those configured with Layer 2 VPNs, Layer 2 circuits, and VPLS routing instances. However, CE Ethernet switches that generate proprietary BPDU frames might not be able to run STP across Juniper Networks routing platforms configured for these emulated Layer 2 connections.

NOTE: Under certain circumstances, VPLS provider routers might duplicate an Internet Control Message Protocol (ICMP) reply from a CE router when a PE router has to flood an ICMP request because the destination MAC address has not yet been learned. The duplicate ICMP reply can be triggered when a CE router with promiscuous mode enabled is connected to a PE router. The PE router automatically floods the promiscuous mode–enabled CE router, which then returns the ICMP request to the VPLS provider routers. The VPLS provider routers consider the ICMP request to be new and flood the request again, creating a duplicate ping reply.

BGP Signaling for VPLS PE Routers Overview

BGP can autonomously signal pseudowires between the PE routers participating in the same virtual private LAN service (VPLS) network. As PE routers are added to and removed from the VPLS network, BGP can signal pseudowires to new PE routers and tear down old pseudowires to old PE routers. Each PE router only needs to be configured with the identity of the VPLS routing instance. Each PE router does not need to be configured with the identities of all of the PE routers that are or might become a part of the VPLS network.

There are three primary ways to signal a virtual private LAN service (VPLS) VPN in an MPLS network. You can power the VPLS using the BGP control plane, or using the LDP control plane, or using a method called FEC 129 which combines both methods of signaling.

One advantage of the BGP control plane is the ability for a provider edge (PE) router to automatically discover any other PE routers that take part in a given VPLS VPN. This is possible thanks to the exchange of route target information that identifies whether a router participates in a particular VPN.

The result is that BGP can autonomously signal pseudowires between the PE routers participating in the same VPLS. This is in contrast to an LDP-signaled VPLS, where remote PEs must be defined explicitly.

As PE routers are added to and removed from the VPLS network, BGP can signal pseudowires to new PE routers. Similarly, BGP can tear down old pseudowires to old PE routers where the VPLS instance has been removed. Each PE router only needs to be configured with the relevant route target information of the VPLS routing instance. There is no need to configure neighbor PEs explicitly.

Note that all of this is completely independent of the choice of protocol to signal the transport label-switched paths (LSPs) that carry the VPN packets between PEs. For example, you are free to choose LDP, RSVP, SR-MPLS, or static LSPs. The choice of transport LSP protocol is entirely separate from the choice of VPLS signaling protocol. This is true of all methods of VPLS signaling.

RELATED DOCUMENTATION

VPLS Multihoming Overview | **885**

VPLS Path Selection Process for PE Routers | **753**

Control Word for BGP VPLS Overview

In a BGP VPLS network, transit routers must determine the payload for hash calculations for load balancing. While parsing an MPLS encapsulated packet for hashing, a transit router can incorrectly calculate an Ethernet payload as an IPv4 or IPv6 payload if the first nibble of the destination address MAC is 0x4 or 0x6, respectively. This false positive can cause out-of-order packet delivery over a pseudowire. This issue can be avoided by configuring a BGP VPLS edge (VE) router to request that other BGP VE routers insert a control word between the label stack and the MPLS payload.

By inserting a control word between the label stack and the Layer 2 header of a packet traversing a VPLS, the first nibble of the destination address MAC can be ensured to be 0, thus preventing the packet from being identified as an IPv4 or IPv6 packet. All VE routers should want incoming packets to contain control words.

BGP is used to negotiate the support for control words between VE routers. You configure a VE router with the `control-word` parameter to indicate the preference to receive packets with the control word. By setting the control word, the VE router expects that all frames marked with a label from the VPLS contain the control word. When remote VE routers advertise their NLRI, if the control word is set on them as well, both ends of the pseudowire have control-word support and the control word is expected in packets arriving at the VE routers in both directions.

If a VE router doesn't have the control word set, a VE router that does have a control word set will act as if the VE router without the control word can neither send nor accept BGP VPLS packets with a control word included.

RELATED DOCUMENTATION

Configuring a Control Word for BGP VPLS | **626**

control-word

Configuring a Control Word for BGP VPLS

In a BGP VPLS network, transit routers must determine the payload for hash calculations for load balancing. While parsing an MPLS encapsulated packet for hashing, a transit router can incorrectly calculate an Ethernet payload as an IPv4 or IPv6 payload if the first nibble of the destination address MAC is 0x4 or 0x6, respectively. This false positive can cause out-of-order packet delivery over a pseudowire. This issue can be avoided by configuring a BGP VPLS PE router to request that other BGP VPLS edge (VE) routers insert a control word between the label stack and the MPLS payload.

WARNING: If you attempt to set a control word in a VPLS network that contains a VE router that does not support a control word, the pseudowire will not come up. To ensure that the pseudowire comes up, be sure that all VE routers in the VPLS network support the presence of a control word.

Before configuring support for a control word in a BGP VPLS network, ensure that the router meets the following requirements:

- MX Series router running Junos OS Release 14.1 or later and containing one of the following Flexible PIC Concentrators: ADPC, NPC, or I-Chip

  OR

- M320 router running Junos OS Release 14.1 or later and containing either the I-Chip or the Trio

To configure a VE router to expect a control word between the label stack and the MPLS payload:

1. At the [edit routing-instances] hierarchy level in configuration mode, set control-word for the VPLS protocol for the specified routing instance.

   ```
   [edit routing-instances]
   user@host# set routing-instance-name protocols vpls control-word
   ```

   For example:

   ```
   [edit routing-instances]
   user@host# set vpls1 protocols vpls control-word
   ```

2. If you are setting control-word on a Trio-based MPC on an MX Series router, set no-ether-pseudowire to omit the IP payload over the Ethernet pseudowire from the hash key.

   ```
   [edit forwarding-options]
   user@host# set enhanced-hash-key family mpls no-ether-pseudowire
   ```

3. Verify the configuration.

   ```
   [edit routing-instances]
   user@host# show
   vpls1 {
       protocols {
           vpls {
               control-word;
           }
       }
   }
   ```

   ```
   [edit forwarding-options]
   user@host# show
   enhanced-hash-key {
       family mpls {
           no-ether-pseudowire;
       }
   }
   ```

4. Repeat the configuration on each VE router in the BGP VPLS network.

5. Run `show vpls connections instance` *`routing-instance-name`* `extensive` to verify the presence of a control word for the pseudowire.

   For example:

```
user@host# show vpls connections instance vpls1 extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
...
PF -- Profile parse failure      PB -- Profile busy

Legend for interface status
Up -- operational
Dn -- down

Instance: vpls1
...
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     May 21 10:08:34 2013           2
    Remote PE: 192.0.2.0, Negotiated control-word: Yes
```

RELATED DOCUMENTATION

Control Word for BGP VPLS Overview | 625

control-word

no-control-word

BGP Route Reflectors for VPLS

In large networks, it might be necessary to configure BGP route reflectors to reduce the control plane workload for the routers participating in the VPLS network. BGP route reflectors can help to reduce the workload of the network control plane in the following ways.

- Making it unnecessary to configure all of the VPLS PE routers in a full mesh.
- Limiting the total volume of BGP VPLS messages exchanged within the network by transmitting messages to interested routers only (instead of all of the BGP routers in the network)

- Reducing the network signaling load whenever another BGP router is added to or removed from the network

The basic solution to these problems is to deploy a small group of BGP route reflectors that are in a full mesh with one another. Each of the VPLS PE routers is configured to have a BGP session with one or more of the route reflectors, making it unnecessary to maintain a full mesh of BGP sessions between all of the PE routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

This type of configuration only affects the control plane of the VPLS network (how routers signal and tear down pseudowires to one another in the network). The actual data plane state and forwarding paths for the VPLS traffic are not modified by the route reflectors. Effectively, the VPLS pseudowires should take the same paths across the network whether or not you have configured route reflectors. For a description of how VPLS selects the best path to a PE router, see "VPLS Path Selection Process for PE Routers" on page 753.

The MAC addresses themselves are not exchanged or processed in any way by BGP. Each VPLS PE router performs all MAC address learning and aging individually. BGP's only function relative to VPLS is to exchange messages related to automatic discovery of PE routers being added to and removed from the VPLS network and the MPLS label exchange needed to signal a pseudowire from one PE router to another.

RELATED DOCUMENTATION

VPLS Path Selection Process for PE Routers | **753**

Example: Configuring a Route Reflector

Example: NG-VPLS Using Point-to-Multipoint LSPs | **1034**

Example: Next-Generation VPLS for Multicast with Multihoming | **1000**

Interoperability Between BGP Signaling and LDP Signaling in VPLS

IN THIS SECTION

- LDP-Signaled and BGP-Signaled PE Router Topology | **630**

- Flooding Unknown Packets Across Mesh Groups | **632**
- Unicast Packet Forwarding | **632**

You can configure a VPLS routing instance where some of the PE routers use BGP for signaling and some use LDP for signaling.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The following concepts form the basis of the configuration needed to include both BGP-signaled and LDP-signaled PE routers in a VPLS routing instance:

- PE router mesh group—Consists of a set of routers participating in a VPLS routing instance that share the same signaling protocol, either BGP or LDP, and are also fully meshed. Each VPLS routing instance can have just one BGP mesh group. However, you can configure multiple LDP mesh groups for each routing instance.

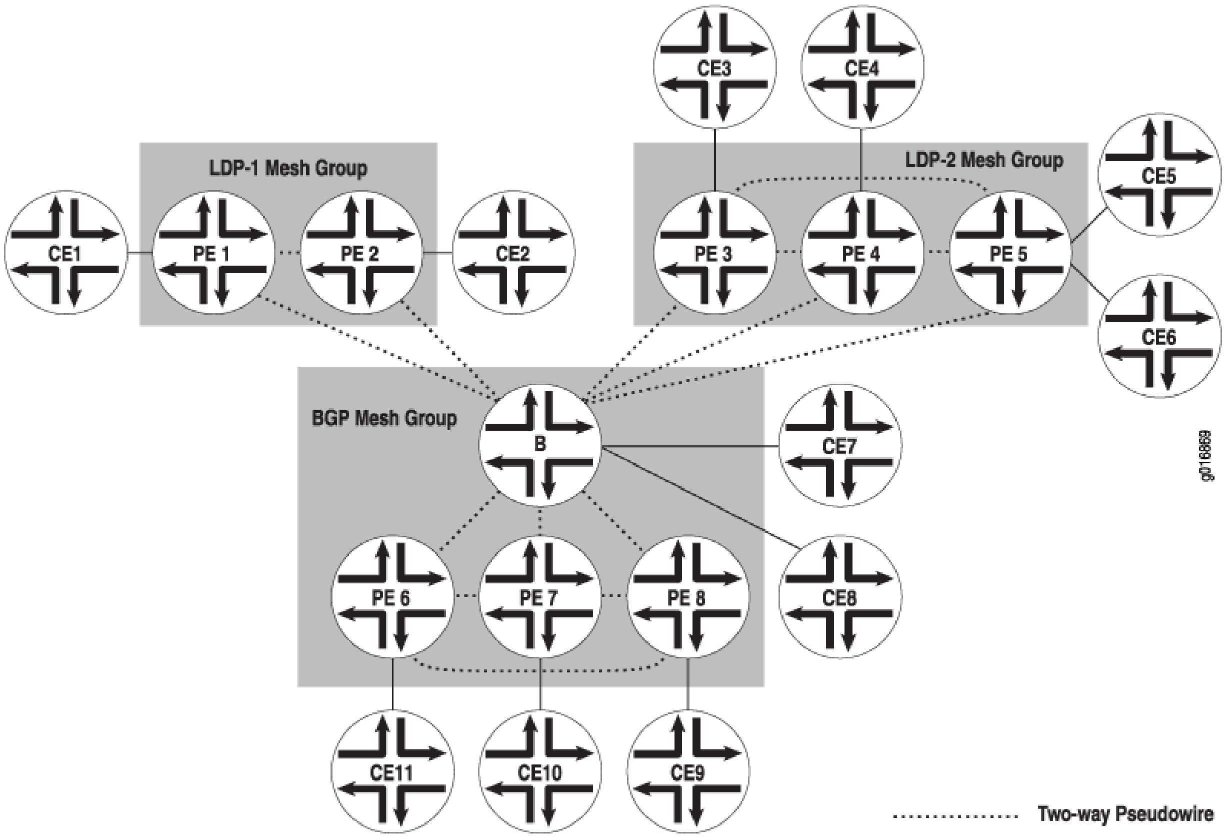
- Border router—A PE router that must be reachable by all of the other PE routers participating in a VPLS routing instance, whether they are LDP-signaled or BGP-signaled. Bidirectional pseudowires are created between the border router and all of these PE routers. The border router is aware of the composition of each PE mesh group configured as a part of the VPLS routing instance. It can also have direct connections to local CE routers, allowing it to act as a typical PE router in a VPLS routing instance.

The following sections describe how the LDP-signaled and BGP-signaled PE routers function when configured to interoperate within a VPLS routing instance:

LDP-Signaled and BGP-Signaled PE Router Topology

Figure 50 on page 631 illustrates a topology for a VPLS routing instance configured to support both BGP and LDP signaling. Router B is the border router. Routers PE1 and PE2 are in the LDP-signaled mesh group LDP-1. Routers PE3, PE4, and PE5 are in the LDP-signaled mesh group LDP-2. Routers PE6, PE7, PE8, and router B (the border router) are in the BGP-signaled mesh group. The border router also acts as a standard VPLS PE router (having local connections to CE routers). All of the PE routers shown are within the same VPLS routing instance.

Figure 50: BGP and LDP Signaling for a VPLS Routing Instance



Two-way pseudowires are established between the PE routers in each mesh group and between each PE router in the VPLS routing instance and the border router. In Figure 50 on page 631, two-way pseudowires are established between routers PE1 and PE2 in mesh group LDP-1, routers PE3, PE4, and PE5 in mesh group LDP-2, and routers PE6, PE7, and PE8 in the BGP mesh group. Routers PE1 through PE8 also all have two-way pseudowires to the Border router. Based on this topology, the LDP-signaled routers are able to interoperate with the BGP-signaled routers. Both the LDP-signaled and BGP-signaled PE routers can logically function within a single VPLS routing instance.

NOTE: The following features are not supported for VPLS routing instances configured with both BGP and LDP signaling:

- Point-to-multipoint LSPs
- Integrated routing and bridging
- IGMP snooping

Flooding Unknown Packets Across Mesh Groups

Broadcast, multicast, and unicast packets of unknown origin received from a PE router are flooded to all local CE routers. They are also flooded to all of the PE routers in the VPLS routing instance except the PE routers that are a part of the originating PE router mesh group.

For example, if a multicast packet is received by the border router in Figure 50 on page 631, it is flooded to the two local CE routers. It is also flooded to routers PE1 and PE2 in the LDP-1 mesh group and to routers PE3, PE4, and PE5 in the LDP-2 mesh group. However, the packet is not flooded to routers PE6, PE7, and PE8 in the BGP mesh group.

Unicast Packet Forwarding

The PE border router is made aware of the composition of each PE router mesh group. From the data plane, each PE router mesh group is viewed as a virtual pseudowire LAN. The border router is configured to interconnect all of the PE router mesh groups belonging to a single VPLS routing instance. To interconnect the mesh groups, a common MAC table is created on the border router.

Unicast packets originating within a mesh group are dropped if the destination is another PE router within the same mesh group. However, if the destination MAC address of the unicast packet is a PE router located in a different mesh group, the packet is forwarded to that PE router.

Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS

IN THIS SECTION

- LDP BGP Interworking Platform Support | **633**
- Configuring FEC 128 VPLS Mesh Groups for LDP BGP Interworking | **633**
- Configuring FEC 129 VPLS Mesh Groups for LDP BGP Interworking | **634**
- Configuring Switching Between Pseudowires Using VPLS Mesh Groups | **634**
- Configuring Integrated Routing and Bridging Support for LDP BGP Interworking with VPLS | **635**
- Configuring Inter-AS VPLS with MAC Processing at the ASBR | **635**

A single VPLS routing instance can encompass one set of PE routers that use BGP for signaling and another set of PE routers that use LDP for signaling. Within each set, all of the PE routers are fully meshed in both the control and data planes and have a bidirectional pseudowire to each of the other routers in the set. However, the BGP-signaled routers cannot be directly connected to the LDP-signaled routers. To be able to manage the two separate sets of PE routers in a single VPLS routing instance, a border PE router must be configured to interconnect the two sets of routers.

The VPLS RFCs and Internet drafts require that all of the PE routers participating in a single VPLS routing instance must be fully meshed in the data plane. In the control plane, each fully meshed set of PE routers in a VPLS routing instance is called a PE router mesh group. The border PE router must be reachable by and have bidirectional pseudowires to all of the PE routers that are a part of the VPLS routing instance, both the LDP-signaled and BGP-signaled routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

For LDP BGP interworking to function, LDP-signaled routers can be configured with forwarding equivalence class (FEC) 128 or FEC 129.

The following sections describe how to configure BGP LDP interworking for VPLS:

LDP BGP Interworking Platform Support

Use Feature Explorer to identify which platforms and releases support LDP BGB interworking in VPLS or any other Junos OS feature.

Configuring FEC 128 VPLS Mesh Groups for LDP BGP Interworking

To configure FEC 128 LDP BGP interworking for VPLS, include the `mesh-group` statement in the VPLS routing instance configuration of the PE border router:

```
mesh-group mesh-group-name {
    local-switching;
    mac-flush [ explicit-mac-flush-message-options ];
    neighbor address;
    peer-as all;
    vpls-id number;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols vpls]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols vpls]`

Using the `neighbor` statement, configure each PE router that is a part of the mesh group. You must separate the LDP-signaled routers and the BGP-signaled routers into their own respective mesh groups. The LDP-signaled routers can be divided into multiple mesh groups. The BGP-signaled routers must be configured within a single mesh group for each routing instance.

Configuring FEC 129 VPLS Mesh Groups for LDP BGP Interworking

Configuration for a mesh group for FEC 129 is very similiar to the configuration for FEC 128.

Note the following differences for FEC 129:

- Each user-defined mesh group must have a unique route distinguisher. Do not use the route distinguisher that is defined for the default mesh group at the `[edit routing-intances]` hierarchy level.
- Each user-defined mesh group must have its own import and export route target.
- Each user-defined mesh group can have a unique Layer 2 VPN ID. By default, all the mesh groups that are configured for the a VPLS routing-instance use the same Layer 2 VPN ID, the one that you configure at the `[edit routing-instances]` hierarchy level.

Configuring Switching Between Pseudowires Using VPLS Mesh Groups

To configure switching between Layer 2 circuit pseudowires using VPLS mesh groups, you can do either of the following:

- Configure a mesh group for each Layer 2 circuit pseudowire terminating at a VPLS routing instance. The Junos OS can support up to 16 mesh groups on MX Series routers and up to 128 on M Series and T Series routers. However, two mesh groups are created by default, one for the CE routers and one for the PE routers. Therefore, the maximum number of user-defined mesh groups is 14 for MX Series routers and 126 for M Series and T Series routers.
- Configure a single mesh group, terminate all the Layer 2 circuit pseudowires into it, and enable local switching between the pseudowires by including the `local-switching` statement at the `[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]` hierarchy level. By default, you cannot configure local switching for mesh groups (except for the CE mesh group) because all of the VPLS PE routers must be configured in a full mesh. However, local switching is useful if you are terminating Layer 2 circuit pseudowires in a mesh group configured for an LDP signaled VPLS routing instance.

NOTE: Do not include the `local-switching` statement on PE routers configured in a full mesh VPLS network.

To terminate multiple pseudowires at a single VPLS mesh group, include the `local-switching` statement:

```
local-switching;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`*`]`

Configuring Integrated Routing and Bridging Support for LDP BGP Interworking with VPLS

Beginning with Junos OS Release 9.4, you can configure an integrated routing and bridging (IRB) interface on a router that functions as an autonomous system border router (ASBR) in an inter-AS VPLS environment between BGP-signaled VPLS and LDP-signaled VPLS. Previously, IRB interfaces were supported only on Provider Edge (PE) routers.

NOTE: ACX Series routers do not support configuring IRB for LDP BGP Interworking with VPLS.

To configure a IRB support for LDP BGP Interworking with VPLS, include the `routing-interface` *`interface-name`* statement.

You can include this statement at the following hierarchy levels:

-
  ```
  [edit routing-instances routing-instance-name]
  ```

-
  ```
  [edit logical-routers logical-router-name routing-instances routing-instance-name]
  ```

Configuring Inter-AS VPLS with MAC Processing at the ASBR

Inter-AS VPLS with MAC processing at the ASBR enables you to interconnect customer sites that are located in different ASs. In addition, you can configure the ASs with different signaling protocols. You can configure one of the ASs with BGP-signaled VPLS and the other with LDP-signaled VPLS. For more information about how to configure LDP-signaled and BGP signaled VPLS, see "Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS" on page 632.

For inter-AS VPLS to function properly, you need to configure IBGP peering between the PE routers, including the ASBRs in each AS, just as you do for a typical VPLS configuration. You also need to configure EBGP peering between the ASBRs in the separate ASs. The EBGP peering is needed between the ASBRs only. The link between the ASBR routers does not have to be Ethernet. You can also connect a CE router directly to one of the ASBRs, meaning you do not have to have a PE router between the ASBR and the CE router.

The configuration for the connection between the ASBRs makes inter-AS VPLS with MAC operations unique. The other elements of the configuration are described in other sections of this manual.

The following sections describe how to configure inter-AS VPLS with MAC operations:

Inter-AS VPLS with MAC Operations Configuration Summary

This section provides a summary of all of the elements which must be configured to enable inter-AS VPLS with MAC operations. These procedures are described in detail later in this chapter and in other parts of theJunos OS VPNs Library for Routing Devices.

The following lists all of major elements of an inter-AS VPLS with MAC operations configuration:

- Configure IBGP between all of the routers within each AS, including the ASBRs.
- Configure EBGP between the ASBRs in the separated ASs. The EBGP configuration includes the configuration that interconnects the ASs.
- Configure a full mesh of LSPs between the ASBRs.
- Configure a VPLS routing instance encompassing the ASBR routers. The ASBRs are VPLS peers and are linked by a single pseudowire. Multihoming between ASs is not supported. A full mesh of pseudowires is needed between the ASBR routers in all of the interconnected ASs.
- Configure the VPLS routing instances using either BGP signaling or LDP signaling. LDP BGP interworking is supported for inter-AS VPLS with MAC operations, so it is possible to interconnect the BGP-signaled VPLS routing instances with the LDP-signaled VPLS routing instances.
- Configure a single VPLS mesh group for all of the ASBRs interconnected using inter-AS VPLS.

Configuring the ASBRs for Inter-AS VPLS

This section describes the configuration on the ASBRs needed to enable inter-AS VPLS with MAC operations.

On each ASBR, you need to configure a VPLS mesh group within the VPLS routing instance which needs to include all of the PE routers within the AS, in addition to the ASBR. You need to configure the same mesh group for each of the ASs you want to interconnect using inter-AS VPLS. The mesh group name should be identical on each AS. You also must include the `peer-as all` statement. This statement enables the router to establish a single pseudowire to each of the other ASBRs.

To configure the mesh group on each ASBR, include the `mesh-group` and `peer-as all` statements:

```
mesh-group mesh-group-name {
    peer-as all;
}
```

You can include these statements at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols vpls]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols vpls]`

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | 913

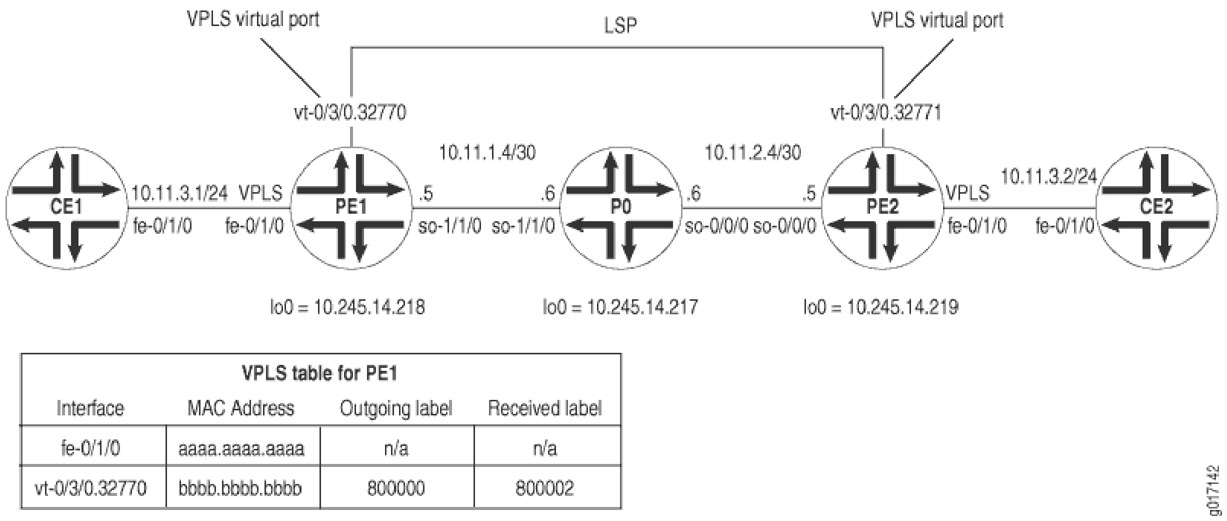
Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938

Example: VPLS Configuration (BGP Signaling)

IN THIS SECTION

● Verifying Your Work | 647

Figure 51: VPLS Topology Diagram



VPLS table for PE1

Interface	MAC Address	Outgoing label	Received label
fe-0/1/0	aaaa.aaaa.aaaa	n/a	n/a
vt-0/3/0.32770	bbbb.bbbb.bbbb	800000	800002

In Figure 51 on page 638, a simple VPLS topology is enabled between routers PE1 and PE2. CE routers CE1 and CE2 use Ethernet-based interfaces to connect VLAN 600 to their local PE router. The PE routers PE1 and PE2 are connected to one another by LSPs enabled across a service provider backbone running MPLS, BGP, RSVP, and OSPF.

In a VPLS routing instance named `green`, PE1 has a local interface `ge-0/1/0` and a virtual port of `vt-0/3/0.32770` (the virtual port is created dynamically on the Tunnel Services PIC when VPLS is configured). PE2 has a local interface `ge-0/1/0` and a virtual port of `vt-0/3/0.32771` in the same `green` instance. As a result, routers CE1 and CE2 send Ethernet traffic to one another as if they were physically connected to each other on a LAN.

On Router CE1, the only item you need to configure is the interface that connects to PE1. Be sure to write down the VLAN identifier and IP address, so you can match them later on CE2.

Router CE1

```
[edit]
interfaces {
    ge-0/1/0 {
        vlan-tagging; # Configure VLAN tagging for VLAN VPLS or extended VLAN VPLS.
        unit 0 {
            vlan-id 600; # The Ethernet interface on CE2 must use the same VLAN ID.
            family inet {
                address 10.11.3.1/24; # The interface on CE2 must use the same prefix.
            }
        }
    }
}
```

If Router PE1 is an MX Series device, you need to configure a tunnel service interface.

To create tunnel interfaces on an MX Series router, include the `tunnel-services` statement at the `[edit chassis fpc` *`slot-number`* `pic` *`number`*`]` hierarchy level. To configure the bandwidth for a tunnel interface, include the `bandwidth` statement at the `[edit chassis fpc` *`slot-number`* `pic` *`number`* `tunnel services]` hierarchy level.

The following example shows a tunnel interface with 1 Gbps of bandwidth configured on PFE 3 of the DPC installed in slot 0 of an MX Series router:

```
[edit chassis]
fpc 0 {
    pic 3 {
        tunnel services {
            bandwidth 1g;
        }
    }
}
```

On Router PE1, prepare the router for VPLS by configuring BGP, MPLS, OSPF, and RSVP. (These protocols are the basis for most Layer 2 VPN-related applications, including VPLS.) Include the `signaling` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level, because VPLS uses the same infrastructure for internal BGP as Layer 2 VPNs.

NOTE: The `signaling` statement replaces the `unicast` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level. You must use the `signaling` statement if you wish to configure VPLS domains and Layer 2 VPNs simultaneously.

Next, configure VLAN tagging on the interface connected to Router CE1. Include VLAN VPLS encapsulation at both the physical and logical interface levels. Be sure to use the same VLAN ID for all Ethernet interfaces that are part of a single VPLS instance. Finally, add the interface into a VPLS routing instance and specify the site range, site ID number, and site name.

Router PE1

```
[edit]
interfaces {
                     ge-0/1/0 {
                         vlan-tagging;# Configure VLAN tagging for VLAN VPLS or extended VLAN
VPLS.
                         encapsulation vlan-vpls; # Configure VPLS encapsulation on both the
        unit 0 { # physical interface and the logical interface.
                               encapsulation vlan-vpls;
                               vlan-id 600;# The VLAN ID is the same one used by the CE routers.
        }
    }
    so-1/1/0 {
        unit 0 {
            family inet {
                address 10.11.1.5/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.245.14.218/32;
            }
        }
    }
}
routing-options {
    autonomous-system 69;
    forwarding-table {
        export exp-to-fwd; # Apply a policy that selects an LSP for the VPLS instance.
    }
}
protocols {
                     rsvp {
        interface all {
            aggregate;
        }
    }
                     mpls {
                           label-switched-path pe1-to-pe2 { # Configure an LSP to reach other 
VPLS PEs.
            to 10.245.14.219;
        }
        interface all;
    }
    bgp {
        group vpls-pe {
            type internal;
            local-address 10.245.14.218;
            family l2vpn { # VPLS uses the same infrastructure as Layer 2 VPNs
                signaling; # for internal BGP.
            }
            neighbor 10.245.14.217;
            neighbor 10.245.14.219;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface so-1/1/0.0 {
                metric 11;
            }
            interface lo0.0 {
                passive;
            }
        }
    }
}
policy-options {
    policy-statement exp-to-fwd {
        term a {
            from community grn-com; # Matches the community in the VPLS instance.
            then {
                install-nexthop lsp pe1-to-pe2; # If there are multiple LSPs that exist
                accept; # between VPLS PE routers, this statement sends VPLS traffic
            } # over a specific LSP.
        }
    }
    community grn-com members target:11111:1; # Adds the instance to a BGP
} # community.
routing-instances {
    green {
        instance-type vpls; # Configure a VPLS routing instance.
        interface ge-0/1/0.0;
        route-distinguisher 10.245.14.218:1;
        vrf-target target:11111:1; # This value is important to the BGP community.
        protocols {
            vpls { # Configure a VPLS site range, site name, and site identifier.
                site-range 10;
                site greenPE1 {
                    site-identifier 1;
                }
            }
        }
    }
}
```

On Router P0, configure BGP, MPLS, OSPF, and RSVP to interconnect PE1 and PE2.

Router P0

```
[edit]
interfaces {
    so-0/0/0 {
        unit 0 {
            family inet {
                address 10.11.2.6/30;
            }
            family mpls;
        }
    }
                    so-1/1/0 {
        unit 0 {
            family inet {
                address 10.11.1.6/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.245.14.217/32;
            }
        }
    }
}
routing-options {
    autonomous-system 69;
}
protocols {
                    rsvp {
        interface all {
            aggregate;
        }
    }
                    mpls {
        interface all;
    }
    bgp {
        group vpls-pe {
            type internal;
            local-address 10.245.14.217;
            family l2vpn { # VPLS uses the same infrastructure as Layer 2 VPNs
                signaling; # for internal BGP.
            }
            neighbor 10.245.14.218;
            neighbor 10.245.14.219;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface so-1/1/0.0 {
                metric 11;
            }
            interface so-0/0/0.0 {
                metric 15;
            }
            interface lo0.0 {
                passive;
            }
        }
    }
}
```

If Router PE2 is an MX Series device, you need to configure a tunnel service interfaces.

To create tunnel interfaces on an MX Series router, include the `tunnel-services` statement at the [`edit chassis fpc` *slot-number* `pic` *number*] hierarchy level. To configure the bandwidth for a tunnel interface, include the `bandwidth` statement at the [`edit chassis fpc` *slot-number* `pic` *number*] hierarchy level.

The following example shows a tunnel interface with 1 Gbps of bandwidth configured on PFE 3 of the DPC installed in slot 0 of an MX Series router:

```
[edit chassis]
fpc 0 {
    pic 3 {
        tunnel services {
            bandwidth 1g;
        }
    }
}
```

On Router PE2, configure BGP, MPLS, OSPF, and RSVP to complement the configuration on PE1. Next, configure VLAN tagging on the interface connected to Router CE2. Include VLAN VPLS encapsulation at both the physical and logical interface levels. Be sure to use the same VLAN ID for all Ethernet interfaces that are part of a single VPLS instance. Finally, add the interface into a VPLS routing instance and specify the site range, site ID number, and site name.

Router PE2

```
[edit]
interfaces {
    ge-0/1/0 {
        vlan-tagging; # Configure VLAN tagging for VLAN VPLS or extended VLAN VPLS.
        encapsulation vlan-vpls; # Configure VPLS encapsulation on both the
        unit 0 { # physical interface and logical interface.
            encapsulation vlan-vpls;
            vlan-id 600;# The VLAN ID is the same one used by the CE routers.
        }
    }
    so-0/0/0 {
        unit 0 {
            family inet {
                address 10.11.2.5/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.245.14.219/32;
            }
        }
    }
}
routing-options {
    autonomous-system 69;
    forwarding-table {
        export exp-to-fwd; # Apply a policy that selects an LSP for the VPLS instance.
    }
}
protocols {
    rsvp {
        interface all {
            aggregate;
        }
    }
    mpls {
        label-switched-path pe2-to-pe1 { # Configure an LSP to other VPLS PE routers.
```

```
            to 10.245.14.218;
        }
        interface all;
    }
    bgp {
        group vpls-pe {
            type internal;
            local-address 10.245.14.219;
            family l2vpn { # VPLS uses the same infrastructure as Layer 2 VPNs
                signaling; # for internal BGP.
            }
            neighbor 10.245.14.217;
            neighbor 10.245.14.218;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface so-0/0/0.0 {
                metric 15;
            }
            interface lo0.0 {
                passive;
            }
        }
    }
}
policy-options {
    policy-statement exp-to-fwd {
        term a {
            from community grn-com; # Matches the community with the VPLS instance.
            then {
                install-nexthop lsp pe2-to-pe1; # If there are multiple LSPs that exist
                accept; # between VPLS PE routers, this statement sends VPLS traffic
            }  # over a specific LSP.
        }
    }
    community grn-com members target:11111:1; # This adds the instance into a BGP community.
}
routing-instances {
    green {
        instance-type vpls; # Configure a VPLS routing instance.
        interface ge-0/1/0.0;
        route-distinguisher 10.245.14.219:1;
        vrf-target target:11111:1; # This value is important for the BGP community.
        protocols {
            vpls { # Configure a VPLS site range, site name, and site identifier.
                site-range 10;
                site greenPE2 {
                    site-identifier 2;
                }
            }
        }
    }
}
```

On Router CE2, complete your VPLS network by configuring the interface that connects to PE2. Use the same VLAN identifier and IP address prefix used on Router CE1.

Router CE2

```
[edit]
interfaces {
    ge-0/1/0 {
        vlan-tagging; # Configure VLAN tagging for VLAN VPLS or extended VLAN VPLS.
        unit 0 {
            vlan-id 600; # The Ethernet interface on CE1 must use the same VLAN ID.
            family inet {
                address 10.11.3.2/24; # The interface on CE1 must use the same prefix.
            }
        }
    }
}
```

Verifying Your Work

To verify proper operation of VPLS, use the following commands:

- `clear vpls mac-address instance` *`instance-name`*
- `show interfaces terse`
- `show route forwarding-table family mpls`
- `show route forwarding-table family vpls (destination | extensive | matching | table)`

- `show route instance (detail)`
- `show system statistics vpls`
- `show vpls connections`
- `show vpls statistics`

The following section shows the output of these commands on Router PE1 as a result of the configuration example:

```
user@PE1> show interfaces terse
Interface               Admin Link Proto Local                 Remote
so-1/1/0                up    up
so-1/1/0.0              up    up   inet  10.11.1.5/30
                                   mpls
so-1/1/1                up    up
so-1/1/2                up    up
so-1/1/3                up    up
ge-0/1/0                up    up
ge-0/1/0.0              up    up   vpls   # This is the local interface
# interface.
ge-0/1/1                up    up
ge-0/1/2                up    up
ge-0/1/3                up    up
gr-0/3/0                up    up
ip-0/3/0                up    up
mt-0/3/0                up    up
pd-0/3/0                up    up
pe-0/3/0                up    up
vt-0/3/0                up    up
vt-0/3/0.32770          up    up  # This is the dynamically generated virtual port.
dsc                     up    up
fxp0                    up    up
fxp0.0                  up    up   inet  192.186.14.218/24
fxp1                    up    up
fxp1.0                  up    up   tnp   4
gre                     up    up
ipip                    up    up
lo0                     up    up
lo0.0                   up    up   inet  10.245.14.218       --> 0/0
                                         127.0.0.1           --> 0/0
                                   inet6 fe80::2a0:a5ff:fe28:13e0
```

```
                                          feee::10:245:14:218
lsi                     up    up
mtun                    up    up
pimd                    up    up
pime                    up    up
tap                     up    up


user@PE1>  show system statistics vpls
vpls:
        0 total packets received
        0 with size smaller than minimum
        0 with incorrect version number
        0 packets for this host
        0 packets with no logical interface
        0 packets with no family
        0 packets with no route table
        0 packets with no auxiliary table
        0 packets with no corefacing entry
        0 packets with no CE-facing entry
         6 mac route learning requests # This indicates that VPLS is working.
        6 mac routes learnt
        0 mac routes aged
        0 mac routes moved
```

To display VPLS source and destination MAC address accounting information, use the `destination`, `extensive`, `matching`, or `table` option with the `show route forwarding-table family vpls` command. When you analyze the display output, keep in mind the following:

- VPLS MAC address accounting is handled on a per-MAC address basis for each VPLS instance. All information is retrieved from MAC address entries in the MAC address table. VPLS MAC address accounting is performed only on local CE routers.
- The VPLS counters for source and destination MAC addresses increment continuously until the oldest MAC address entries are removed from the memory buffer, either when the entries time out or if the VPLS instance is restarted.

```
user@PE1> show route forwarding-table family vpls extensive
Routing table: green.vpls [Index 2]
VPLS:

Destination:  default
  Route type: dynamic             Route reference: 0
  Flags: sent to PFE
  Next-hop type: flood                 Index: 353      Reference: 1

Destination:  default
  Route type: permanent           Route reference: 0
  Flags: none
  Next-hop type: discard               Index: 298      Reference: 1

Destination:  ge-0/1/0.0
  Route type: dynamic             Route reference: 0
  Flags: sent to PFE
  Next-hop type: flood                 Index: 355      Reference: 1

Destination:  bb:bb:bb:bb:bb:bb/48 # This MAC address belongs to remote CE2.
  Route type: dynamic             Route reference: 0
  Flags: sent to PFE, prefix load balance
  Next-hop type: indirect              Index: 351      Reference: 4
  Next-hop type: Push 800000, Push 100002(top)
  Next-hop interface: so-1/1/0.0

Destination:  aa:aa:aa:aa:aa:aa/48 # This MAC address belongs to local CE1.
  Route type: dynamic             Route reference: 0
  Flags: sent to PFE, prefix load balance
  Next-hop type: unicast               Index: 354      Reference: 2
  Next-hop interface: ge-0/1/0.0

user@PE1>  show route forwarding-table family vpls
Routing table: green.vpls
VPLS:
Destination        Type RtRef Next hop          Type Index NhRef Netif
default            dynm     0                   flood  353     1
default            perm     0                   dscd   298     1
ge-0/1/0.0         dynm     0                   flood  355     1
bb:bb:bb:bb:bb:bb/48 # This MAC address belongs to remote CE2.
                   dynm     0                   indr   351     4
                                                Push 800000, Push 100002(top)
so-1/1/0.0
aa:aa:aa:aa:aa:aa/48 # This MAC address belongs to local CE1.
                   dynm     0                   ucst   354     2 ge-0/1/0.0


user@PE1>  show route forwarding-table family mpls
Routing table: mpls
```

```
MPLS:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    dscd    19     1
0                  user     0                    recv    18     3
1                  user     0                    recv    18     3
2                  user     0                    recv    18     3
100000             user     0 10.11.1.6          swap  100001     so-1/1/0.0
800002             user     0                    Pop              vt-0/3/0.32770
vt-0/3/0.32770 (VPLS)
                   user     0                    indr   351     4
                                                 Push 800000, Push 100002(top) so-1/1/0.0


user@PE1> show route instance green detail
green:
  Router ID: 0.0.0.0
  Type: vpls              State: Active
  Interfaces:
    ge-0/1/0.0  # This is the local interface.
    vt-0/3/0.32770  # This is the dynamically generated VPLS virtual port.
  Route-distinguisher: 10.245.14.218:1
  Vrf-import: [ __vrf-import-green-internal__ ]
  Vrf-export: [ __vrf-export-green-internal__ ]
  Vrf-import-target: [ target:11111:1 ]
  Vrf-export-target: [ target:11111:1 ]
  Tables:
    green.l2vpn.0          : 2 routes (2 active, 0 holddown, 0 hidden)


user@PE1> show vpls connections
L2VPN Connections:
Legend for connection status (St)
OR -- out of range              WE -- intf encaps != instance encaps
EI -- encapsulation invalid     Dn -- down
EM -- encapsulation mismatch    VC-Dn -- Virtual circuit down
CM -- control-word mismatch     -> -- only outbound conn is up
CN -- circuit not present       <- -- only inbound  conn is up
OL -- no outgoing label         Up -- operational
NC -- intf encaps not CCC/TCC   XX -- unknown
NP -- interface not present
Legend for interface status
Up -- operational
Dn -- down
```

```
Instance: green
Local site: greenPE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Jan 24 06:26:49 2003           1
      Local interface: vt-0/3/0.32770, Status: Up, Encapsulation: VPLS
      Remote PE: 10.245.14.219, Negotiated control-word: No
      Incoming label: 800002, Outgoing label: 800000


user@PE1>  show system statistics vpls
vpls:
        0 total packets received
        0 with size smaller than minimum
        0 with incorrect version number
        0 packets for this host
        0 packets with no logical interface
        0 packets with no family
        0 packets with no route table
        0 packets with no auxiliary table
        0 packets with no corefacing entry
        0 packets with no CE-facing entry
        7 mac route learning requests
        7 mac routes learnt
        0 mac routes aged
        0 mac routes moved


user@PE1>  show route instance green detail
green:
  Router ID: 0.0.0.0
  Type: vpls              State: Active
   Interfaces:
    ge-0/1/0.0
    vt-0/3/0.32770
  Route-distinguisher: 10.245.14.218:1
  Vrf-import: [ __vrf-import-green-internal__ ]
  Vrf-export: [ __vrf-export-green-internal__ ]
  Vrf-import-target: [ target:11111:1 ]
  Vrf-export-target: [ target:11111:1 ]
  Tables:
    green.l2vpn.0          : 2 routes (2 active, 0 holddown, 0 hidden)

user@PE1> show vpls statistics
```

```
Layer-2 VPN Statistics:
Instance: green
   Local interface: ge-0/1/0.0, Index: 351
   Remote provider edge router: 10.245.14.219
     Multicast packets:                    363
     Multicast bytes  :                  30956
     Flood packets    :                      0
     Flood bytes      :                      0
   Local interface: vt-0/3/0.32770, Index: 354
   Remote provider edge router: 10.245.14.219
     Multicast packets:                    135
     Multicast bytes  :                  12014
     Flood packets    :                    135
     Flood bytes      :                  12014
```

To clear all MAC address entries for a VPLS instance from the VPLS table, issue the `clear vpls mac-address instance` *`instance-name`* command. Add the `logical-system` *`logical-system-name`* option to clear entries in a VPLS instance within a logical system. Use the *`mac-address`* option to remove individual MAC addresses.

RELATED DOCUMENTATION

Virtual Private LAN Services

Example: VPLS Configuration (BGP and LDP Interworking)

IN THIS SECTION

- Verifying Your Work | 668

Figure 52: Topology for VPLS Configuration Example

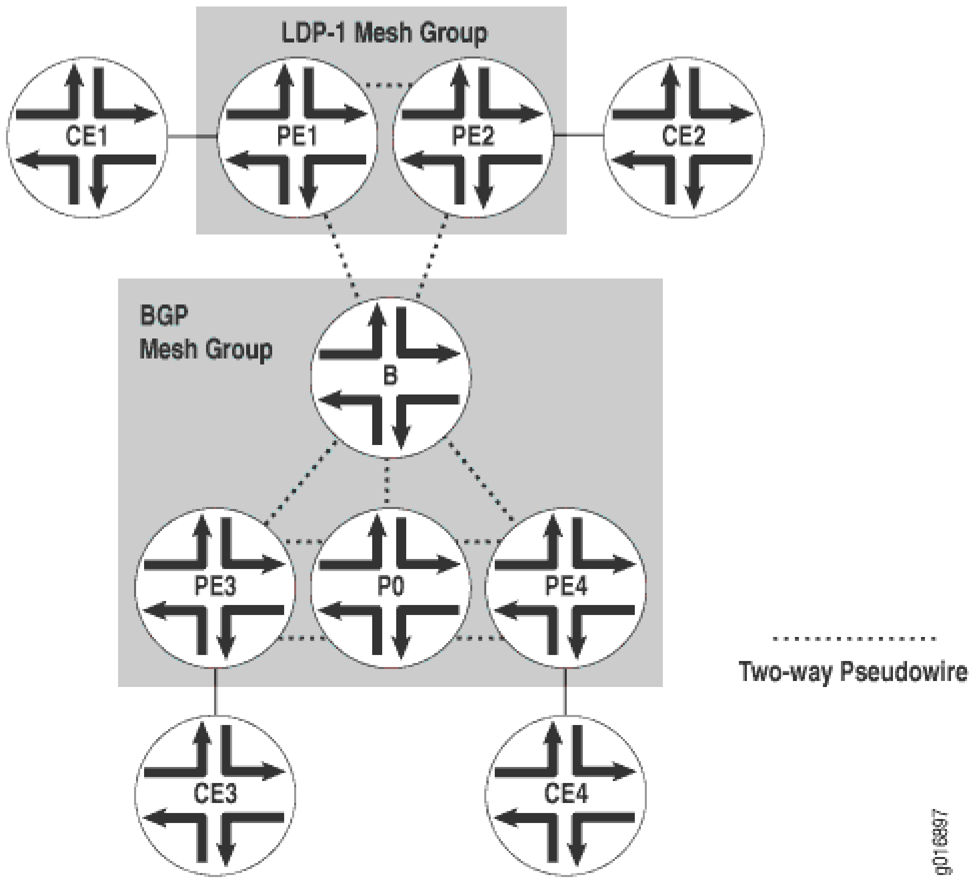


Figure 52 on page 654, shows two VPLS mesh groups: LDP-1 and the default BGP mesh group. The VPLS instance is named v1 in the configuration. Table 15 on page 654 shows the addresses for the router interfaces in the example topology.

Table 15: Router Interface Addresses for VPLS Configuration Example

Router	Interface	Address
CE1	fe-0/0/3 (link to Router PE1)	10.12.31.1
	loopback	10.12.53.1
CE2	fe-0/0/1 (link to Router PE2)	10.12.31.2
	loopback	10.12.53.2

Table 15: Router Interface Addresses for VPLS Configuration Example *(Continued)*

Router	Interface	Address
PE1	t1-1/1/1 (link to Router PE2)	10.12.100.17
	t1-0/1/0 (link to Router B)	10.12.100.2
	loopback	10.255.170.106
PE2	t1-0/1/1 (link to Router PE1)	10.12.100.18
	t1–0/1/3 (link to Router B)	10.12.100.6
	loopback	10.255.170.104
B	t1-0/1/2 (link to Router PE1)	10.12.100.1
	t1-0/1/3 (link to Router PE2)	10.12.100.5
	so-0/2/2 (link to Router PE3)	10.12.100.9
	fe-0/0/3 (link to Router PE4)	10.12.100.13
	loopback	10.255.170.98
PE3	s0-0/2/1 (link to Router B)	10.12.100.10
	so-0/2/2 (link to Router P0)	10.12.100.21
	loopback	10.255.170.96
P0	so-0/2/1 (link to Router PE3)	10.12.100.22

Table 15: Router Interface Addresses for VPLS Configuration Example *(Continued)*

Router	Interface	Address
	t1-0/1/3 (link to Router PE4)	10.12.100.25
	loopback	10.255.170.100
PE4	fe-0/0/3 (link to Router B)	10.12.100.14
	t1-0/1/3 (link to Router P0)	10.12.100.26
	loopback	10.255.170.102
CE3	ge-1/2/1 (link to PE3)	10.12.31.3
	loopback	10.12.53.3
CE4	fe-0/0/2 (link to PE4)	10.12.31.4
	loopback	10.12.53.4

On Router CE3, the only item you need to configure is the Gigabit Ethernet interface that connects to PE3.

Router CE3

```
[edit]
interfaces {
    ge-1/2/1 {
        unit 0 {
            family inet {
                address 10.12.31.1/24;
            }
        }
    }
}
```

On Router PE3, prepare the router for VPLS by configuring BGP, MPLS, OSPF, and LDP. (These protocols are the basis for most Layer 2 VPN-related applications, including VPLS.) Include the `signaling` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level, because VPLS uses the same infrastructure for internal BGP as Layer 2 VPNs.

NOTE: In Junos OS Release 7.3 and later, the `signaling` statement replaces the `unicast` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level. You must use the `signaling` statement if you wish to configure VPLS domains and Layer 2 VPNs simultaneously.

Next, configure VLAN tagging on the Gigabit Ethernet interface connected to Router CE3. Finally, add the Gigabit Ethernet interface into a VPLS routing instance and specify the site range, site ID number, and site name.

Router PE3

```
[edit]
interfaces {
    so-0/2/1 {
        unit 0 {
            family inet {
                address 10.12.100.10/30;
            }
            family mpls;
        }
    }
    so-0/2/2 {
        unit 0 {
            family inet {
                address 10.12.100.21/30;
            }
            family mpls;
        }
    }
    ge-1/3/1 {
        encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    }
}
routing-instances {
    v1 {
        instance-type vpls;
        protocols {
            vpls {
                site 1 {
                    site-identifier 3;
                }
                site-range 10;
            }
        }
        interface ge-1/3/1.0;
        route-distinguisher 10.255.170.96:1;
        vrf-target target:1:2;
    }
}
routing-options {
    autonomous-system 65000;
}
protocols {
    bgp {
        group int {
            type internal;
            local-address 10.255.170.96;
            family l2vpn {
                signaling;
            }
            neighbor 10.255.170.98;
            neighbor 10.255.170.102;
        }
        log-updown;
    }
    ldp {
        interface so-0/2/1.0;
        interface so-0/2/2.0;
    }
    mpls {
        interface all;
    }
    ospf {
        area 0.0.0.0 {
            interface so-0/2/1.0;
            interface so-0/2/2.0;
            interface lo0.0 {
                passive;
            }
        }
    }
}
```

On Router P0, configure MPLS, OSPF, and LDP to interconnect PE3 and PE4.

Router P0

```
[edit]
interfaces {
    t1-0/1/3 {
        unit 0 {
            family inet {
                address 10.12.100.25/30;
            }
            family mpls;
        }
    so-0/2/1 {
        unit 0 {
            family inet {
                address 10.12.100.22/30;
            }
            family mpls;
        }
    }
}
protocols {
    mpls {
        interface all;
    }
    ospf {
        area 0.0.0.0 {
            interface so-0/2/1.0;
            interface t1-0/1/3.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface t1-0/1/3.0;
        interface so-0/2/1.0;
    }
}
```

On Router PE4, configure BGP, MPLS, OSPF, and LDP to complement the configuration on PE3. Next, configure VLAN tagging on the Fast Ethernet interface connected to Router CE4. Include VLAN VPLS encapsulation at both the physical and logical interface levels. Finally, add the Fast Ethernet interface into a VPLS routing instance and specify the site range, site ID number, and site name.

Router PE4

```
[edit]
interfaces {
    fe-0/0/2 {
    encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    fe-0/0/3 {
        unit 0 {
            family inet {
                address 10.12.100.14/30;
            }
            family mpls;
        }
    }
    t1-0/1/3 {
        unit 0 {
            family inet {
                address 10.12.100.26/30;
            }
            family mpls;
            }
        }
    }
}
protocols {
    mpls {
        interface all;
    }
    bgp {
        log-updown;
        group int {
            type internal;
            local-address 10.255.170.102;
            family l2vpn {
                signaling;
            }
            neighbor 10.255.170.96;
            neighbor 10.255.170.98;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface fe-0/0/3.0;
            interface t1-0/1/3.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface fe-0/0/3.0;
        interface t1-0/1/3.0;
        interface lo0.0;
        }
    }
routing-instances {
    v1 {
        instance-type vpls;
        interface fe-0/0/2.0;
        route-distinguisher 10.255.170.102:1;
        vrf-target target:1:2;
        protocols {
            vpls {
                site-range 10;
                site 1 {
                    site-identifier 4;
                }
            }
        }
    }
routing-options {
    autonomous-system 65000;
    }
```

On Router CE4, configure the Fast Ethernet interface that connects to PE4.

Router CE4

```
[edit]
interfaces {
    fe-0/0/2 {
        unit 0 {
            family inet {
                address 10.12.31.4/24;
            }
        }
    }
}
```

On Router B, the area border router, configure the interfaces. Next, configure BGP, MPLS, OSPF, and LDP. Be sure to include the loopback interface in the LDP configuration by including the `interface lo0.0` statement at the [`edit protocols ldp`] hierarchy level. For BGP, include the `signaling` statement at the [`edit bgp group` *`group-name`* `family l2vpn`] hierarchy level. Last, configure the VPLS instance with both BGP and LDP signaling. Configure the LDP-1 mesh group by including the `mesh-group ldp1` statement at the [`edit routing-instances v1 protocols vpls`] hierarchy level.

Router B

```
[edit]
interfaces {
    fe-0/0/3 {
        unit 0 {
            family inet {
                address 10.12.100.13/30;
            }
            family mpls;
        }
    }
    t1-0/1/2 {
        unit 0 {
            family inet {
                address 10.12.100.1/30;
            }
            family mpls;
        }
    }
    t1-0/1/3 {
        unit 0 {
            family inet {
                address 10.12.100.5/30;
            }
            family mpls;
        }
    }
    so-0/2/2 {
        unit 0 {
            family inet {
                address 10.12.100.9/30;
            }
            family mpls;
        }
    }
}
protocols {
    mpls {
        interface all;
    }
    bgp {
        log-updown;
        group int {
            type internal;
            local-address 10.255.170.98;
            family l2vpn {
                signaling;
            }
            neighbor 10.255.170.96;
            neighbor 10.255.170.102;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface t1-0/1/2.0;
            interface t1-0/1/3.0;
            interface so-0/2/2.0;
            interface fe-0/0/3.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface fe-0/0/3.0;
        interface t1-0/1/2.0;
        interface t1-0/1/3.0;
        interface so-0/2/2.0;
        interface lo0.0;
    }
}
routing-instances {
    v1 {
        instance-type vpls;
        route-distinguisher 10.255.170.98:1;
        vrf-target target:1:2;
        protocols {
            vpls {
                site-range 10;
                site 1 {
                    site-identifier 1;
                }
                vpls-id 101;
                mesh-group ldp-1 {
                    neighbor 10.255.170.106;
                    neighbor 10.255.170.104;
                    }
                }
            }
        }
    }
routing-options {
    autonomous-system 65000;
    }
```

Finally, configure the LDP PE routers. On Router PE1, prepare the router for VPLS by configuring LDP, MPLS, and OSPF. Next, configure VPLS encapsulation on the Fast Ethernet interface connected to CE1. Finally, add the Fast Ethernet interface to the routing instance, specifying the VPLS ID and the neighboring routers' loopback addresses.

Router PE1

```
[edit]
interfaces {
    fe-0/0/3 {
        encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    }
    t1-0/1/0 {
        unit 0 {
            family inet {
                address 10.12.100.2/30;
            }
            family mpls;
        }
    }
    t1-1/1/1 {
        unit 0 {
            family inet {
                address 10.12.100.17/30;
            }
            family mpls;
        }
    }
}
protocols {
    mpls {
        interface all;
    }
    ospf {
        area 0.0.0.0 {
            interface t1-0/1/0.0;
            interface t1-1/1/1.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface t1-0/1/0.0;
        interface t1-1/1/1.0;
        interface lo0.0;
    }
}
routing-instances {
    v1 {
        instance-type vpls;
        interface fe-0/0/3.0;
        protocols {
            vpls {
                vpls-id 101;
                neighbor 10.255.170.98;
                neighbor 10.255.170.104;
            }
        }
    }
}
```

Next, configure the Fast Ethernet interface on Router CE1 that connects to Router PE1.

Router CE1

```
[edit]
interfaces {
    fe-0/0/3 {
        unit 0 {
            family inet {
                address 10.12.31.1/24;
            }
        }
    }
}
```

On Router PE2, prepare the router for VPLS by configuring LDP, MPLS, and OSPF. Next, configure VPLS encapsulation on the Fast Ethernet interface connected to Router CE1. Finally, add the Fast Ethernet interface to the routing instance, specifying the VPLS ID and the neighboring routers' loopback addresses.

Router PE2

```
[edit]
interfaces {
    t1-0/1/1 {
        unit 0 {
            family inet {
                address 10.12.100.18/30;
            }
            family mpls;
        }
    t1-0/1/3 {
        unit 0 {
            family inet {
                address 10.12.100.6/30;
            }
            family mpls;
        }
    }
    fe-1/0/2 {
        encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    }
}
protocols {
    mpls {
        interface all;
    }
    ospf {
        area 0.0.0.0 {
            interface t1-0/1/3.0;
            interface t1-0/1/1.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface t1-0/1/1.0;
        interface t1-0/1/3.0;
        interface lo0.0;
    }
}
routing-instances {
    v1 {
        instance-type vpls;
        interface fe-1/0/2.0;
        protocols {
            vpls {
                vpls-id 101;
                neighbor 10.255.170.98;
                neighbor 10.255.170.106;
            }
        }
    }
}
```

Finally, on Router CE2 configure the Fast Ethernet interface connected to PE2:

Router CE2

```
[edit]
interfaces {
    fe-0/0/1 {
        unit 0 {
            family inet {
                address 10.12.31.2/24;
            }
        }
    }
}
```

Verifying Your Work

To verify proper operation of VPLS, use the following commands:

- `show bgp summary`
- `show ldp neighbor`
- `show vpls connections`
- `show route forwarding-table family vpls (destination | extensive | matching | table)`
- `show interfaces vt* terse`
- `show vpls flood extensive`
- `show vpls statistics`

The following section shows the output of some of these commands on Router B as a result of the configuration example.

Use the `show bgp summary` command to verify BGP signaling for VPLS is up.

```
user@B> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0            2          2          0          0          0          0
Peer               AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Damped...
10.255.170.96   65000        124        125       0       0       54:26 Establ
  bgp.l2vpn.0: 1/1/0
  v1.l2vpn.0: 1/1/0
10.255.170.102  65000        122        124       0       0       54:18 Establ
  bgp.l2vpn.0: 1/1/0
  v1.l2vpn.0: 1/1/0
```

Use the `show ldp neighbors` command to verify that LDP signaling for VPLS is up.

```
user@B>  show ldp neighbors
Address            Interface          Label space ID         Hold time
10.255.170.104     lo0.0              10.255.170.104:0         41
10.255.170.106     lo0.0              10.255.170.106:0         38
10.12.100.14       fe-0/0/3.0         10.255.170.102:0         12
10.12.100.10       so-0/2/2.0         10.255.170.96:0          14
10.12.100.2        t1-0/1/2.0         10.255.170.106:0         14
10.12.100.6        t1-0/1/3.0         10.255.170.104:0         13
```

To verify that the VPLS connections are up, use the `show vpls connections` command.

```
user@B>show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
```

```
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision LN -- local site not
designated  LM -- local site ID not minimum designated RN -- remote site not designated RM --
remote site ID not minimum designated XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble

Legend for interface status
Up -- operational
Dn -- down

Instance: v1
BGP-VPLS State
  Local site: 1 (1)
    connection-site           Type  St     Time last up          # Up trans
    3                         rmt   Up     Jan 22 16:38:47 2008           1
      Local interface: vt-0/3/0.1048834, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls v1 local site 1 remote site 3
      Remote PE: 10.255.170.96, Negotiated control-word: No
      Incoming label: 800258, Outgoing label: 800000
    4                         rmt   Up     Jan 22 16:38:54 2008           1
      Local interface: vt-0/3/0.1048835, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls v1 local site 1 remote site 4
      Remote PE: 10.255.170.102, Negotiated control-word: No
      Incoming label: 800259, Outgoing label: 800000 LDP-VPLS State
VPLS-id: 101
  Mesh-group connections: m1
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.170.104(vpls-id 101) rmt Up     Jan 22 16:38:40 2008           1
      Local interface: vt-0/3/0.1048833, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls v1 neighbor 10.255.170.104 vpls-id 101
      Remote PE: 10.255.170.104, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 800000
    10.255.170.106(vpls-id 101) rmt Up     Jan 22 16:38:39 2008           1
      Local interface: vt-0/3/0.1048832, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls v1 neighbor 10.255.170.106 vpls-id 101
      Remote PE: 10.255.170.106, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800000
```

To display VPLS routes (MAC addresses) in the vpls forwarding table, use the `show route forwarding-table family vpls` command.

```
user@B> show route forwarding-table family vpls
Routing table: v1.vpls
VPLS:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   540     1
vt-0/3/0.1048832   user     0                    comp   587     3
vt-0/3/0.1048833   user     0                    comp   587     3
vt-0/3/0.1048834   user     0                    comp   589     3
vt-0/3/0.1048835   user     0                    comp   589     3
00:17:cb:c2:10:01/48
                   dynm     0                    indr 262143     4
                                                Push 800000   580     2
t1-0/1/3.0
00:17:cb:c2:10:02/48
                   dynm     0                    indr 262145     4
                              10.12.100.14      Push 800000   594     2
fe-0/0/3.0
00:17:cb:c2:10:03/48
                   dynm     0                    indr 262142     4
                                                Push 800000   576     2
t1-0/1/2.0
00:17:cb:c2:10:bd/48
                   dynm     0                    indr 262144     4
                                                Push 800000   585     2
so-0/2/2.0
```

To display VPLS source and destination MAC address accounting information, use the `destination`, `extensive`, `matching`, or `table` option with the `show route forwarding-table family vpls` command. When you analyze the display output, keep in mind the following:

- VPLS MAC address accounting is handled on a per-MAC address basis for each VPLS instance. All information is retrieved from MAC address entries in the MAC address table. VPLS MAC address accounting is performed only on local CE routers.
- The VPLS counters for source and destination MAC addresses increment continuously until the oldest MAC address entries are removed from the memory buffer, either when the entries time out or if the VPLS instance is restarted.

To display status information about Virtual Loopback Tunnel interfaces in the VPLS instance, use the `show interfaces vt* terse` command.

```
user@B> show interfaces vt* terse
Interface               Admin Link Proto    Local                 Remote
vt-0/3/0                up    up
vt-0/3/0.1048832        up    up   vpls
vt-0/3/0.1048833        up    up   vpls
vt-0/3/0.1048834        up    up   vpls
vt-0/3/0.1048835        up    up   vpls
```

To display VPLS route information related to the flood process, use the `show vpls flood extensive` command.

```
user@B> show vpls flood extensive
Name: v1
CEs: 0
VEs: 4
  Flood route prefix: 0x4a/32
  Flood route type: IFF_FLOOD
  Flood route owner: vt-0/3/0.1048834
  Flood group name: __ves__
  Flood group index: 0
  Nexthop type: comp
  Nexthop index: 589
    Flooding to:
    Name             Type          NhType          Index
    m1               Group          comp            588
        Composition: flood-to-all
        Flooding to:
        Name             Type          NhType          Index
        vt-0/3/0.1048832 VE             indr            262142
        vt-0/3/0.1048833 VE             indr            262143

  Flood route prefix: 0x4b/32
  Flood route type: IFF_FLOOD
  Flood route owner: vt-0/3/0.1048835
  Flood group name: __ves__
  Flood group index: 0
  Nexthop type: comp
  Nexthop index: 589
    Flooding to:
    Name            Type          NhType          Index
    m1              Group          comp            588
        Composition: flood-to-all
        Flooding to:
        Name            Type          NhType          Index
        vt-0/3/0.1048832 VE            indr            262142
        vt-0/3/0.1048833 VE            indr            262143

  Flood route prefix: 0x48/32
  Flood route type: IFF_FLOOD
  Flood route owner: vt-0/3/0.1048832
  Flood group name: m1
  Flood group index: 2
  Nexthop type: comp
  Nexthop index: 587
    Flooding to:
    Name            Type          NhType          Index
    __ves__         Group          comp            586
        Composition: flood-to-all
        Flooding to:
        Name            Type          NhType          Index
        vt-0/3/0.1048834 VE            indr            262144
        vt-0/3/0.1048835 VE            indr            262145

  Flood route prefix: 0x49/32
  Flood route type: IFF_FLOOD
  Flood route owner: vt-0/3/0.1048833
  Flood group name: m1
  Flood group index: 2
  Nexthop type: comp
  Nexthop index: 587
    Flooding to:
    Name            Type          NhType          Index
    __ves__         Group          comp            586
        Composition: flood-to-all
        Flooding to:
        Name            Type          NhType          Index
        vt-0/3/0.1048834 VE            indr            262144
        vt-0/3/0.1048835 VE            indr            262145
```

To view packet flow statistics for the VPLS instance, use the `show vpls statistics` command:

```
user@B> show vpls statistics
Instance: v1
   Local interface: vt-0/3/0.1048832, Index: 72
   Remote PE: 10.255.170.106
     Multicast packets:                    6
     Multicast bytes  :                  360
     Flooded packets  :                   16
     Flooded bytes    :                 1188
     Current MAC count:                    1
   Local interface: vt-0/3/0.1048833, Index: 73
   Remote PE: 10.255.170.104
     Multicast packets:                    4
     Multicast bytes  :                  240
     Flooded packets  :                    6
     Flooded bytes    :                  398
     Current MAC count:                    1
   Local interface: vt-0/3/0.1048834, Index: 74
   Remote PE: 10.255.170.96
     Multicast packets:                    2
     Multicast bytes  :                  120
     Flooded packets  :                    4
     Flooded bytes    :                  278
     Current MAC count:                    1
   Local interface: vt-0/3/0.1048835, Index: 75
   Remote PE: 10.255.170.102
     Multicast packets:                    1
     Multicast bytes  :                   60
     Flooded packets  :                    2
     Flooded bytes    :                  158
     Current MAC count:                    1
```

RELATED DOCUMENTATION

Virtual Private LAN Services

CHAPTER 33

Assigning Routing Instances to VPLS

IN THIS CHAPTER

- Configuring VPLS Routing Instances | 675
- Configuring a VPLS Routing Instance | 693
- Support of Inner VLAN List and Inner VLAN Range for Qualified BUM Pruning on a Dual-Tagged Interface for a VPLS Routing Instance Overview | 694
- Configuring Qualified BUM Pruning for a Dual-Tagged Interface with Inner VLAN list and InnerVLAN range for a VPLS Routing Instance | 696
- Configuring a Layer 2 Control Protocol Routing Instance | 698
- PE Router Mesh Groups for VPLS Routing Instances | 699
- Configuring VPLS Fast Reroute Priority | 700
- Specifying the VT Interfaces Used by VPLS Routing Instances | 702
- Understanding PIM Snooping for VPLS | 703
- Example: Configuring PIM Snooping for VPLS | 704
- VPLS Label Blocks Operation | 719
- Configuring the Label Block Size for VPLS | 724
- Example: Building a VPLS From Router 1 to Router 3 to Validate Label Blocks | 725

Configuring VPLS Routing Instances

IN THIS SECTION

- Configuring BGP Signaling for VPLS | 677
- Configuring LDP Signaling for VPLS | 683
- Configuring VPLS Routing Instance and VPLS Interface Connectivity | 686
- Configuring the VPLS Encapsulation Type | 687

- Configuring the MPLS Routing Table to Leak Routes a Nondefault Routing Instance | 688
- Configuring the VPLS MAC Table Timeout Interval | 689
- Configuring the Size of the VPLS MAC Address Table | 689
- Limiting the Number of MAC Addresses Learned from an Interface | 690
- Removing Addresses from the MAC Address Database | 692

To configure a VPLS routing instance, include the `vpls` statement:

```
vpls {
    active-interface {
        any;
        primary interface-name;
    }
    connectivity-type (ce | irb | permanent);
    control-word;
    encapsulation-type encapsulation-type;
    interface-mac-limit limit;
    import-labeled-routes [ routing-instance-name ];
    label-block-size size;
    mac-table-aging-time time;
    mac-table-size size;
    neighbor neighbor-id;
    no-control-word;
    no-tunnel-services;
    site site-name {
        active-interface {
            any;
            primary interface-name;
        }
        interface interface-name {
            interface-mac-limit limit;
        }
        mesh-group mesh-group-name;
        multi-homing;
        site-identifier identifier;
        site-preference preference-value {
            backup;
            primary;
        }
    }
    site-range number;
    traceoptions {
        file filename <files number> <size size> <world-readable | no-world-readable>;
        flag flag <flag-modifier> <disable>;
    }
    tunnel-services {
        devices device-names;
        primary primary-device-name;
    }
    vpls-id vpls-id;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

NOTE: You cannot configure a routing protocol (OSPF, RIP, IS-IS or BGP) inside a VPLS routing instance (`instance-type vpls`). The Junos CLI disallows this configuration.

NOTE: Starting in Junos OS Release 16.1, you can use the `import-labeled-routes` statement to specify one or more nondefault routing instances where you want MPLS pseudowire labeled routes to be leaked from the mpls.0 path routing table in the primary routing instance.

The configuration for the VPLS routing instance statements is explained in the following sections:

Configuring BGP Signaling for VPLS

You can configure BGP signaling for the VPLS routing instance. BGP is used to signal the pseudowires linking each of the PE routers participating in the VPLS routing instance. The pseudowires carry VPLS traffic across the service provider's network between the VPLS sites.

NOTE: You cannot configure both BGP signaling and LDP signaling for the same VPLS routing instance. If you attempt to configure the statements that enable BGP signaling for the VPLS routing instance (the `site`, `site-identifier`, and `site-range` statements) and the statements that enable LDP signaling for the same instance (the `neighbor` and `vpls-id` statements), the commit operation fails.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Configure BGP signaling for the VPLS routing instance by completing the steps in the following sections:

Configuring the VPLS Site Name and Site Identifier

When you configure BGP signaling for the VPLS routing instance, on each PE router you must configure each VPLS site that has a connection to the PE router. All the Layer 2 circuits provisioned for a VPLS site are listed as the set of logical interfaces (using the `interface` statement) within the `site` statement.

You must configure a site name and site identifier for each VPLS site.

To configure the site name and the site identifier, include the `site` and the `site-identifier` statements:

```
site site-name {
    interface interface-name {
        interface-mac-limit limit;
    }
    site-identifier identifier;
}
```

The numerical identifier can be any number from 1 through 65,534 that uniquely identifies the local VPLS site.

You can include these statements at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

Configuring Automatic Site Identifiers for VPLS

When you enable automatic site identifiers, the Junos OS automatically assigns site identifiers to VPLS sites. Only one site is allowed per routing instances when using the `automatic-side-id` function. To configure automatic site identifiers for a VPLS routing instance, include the `automatic-site-id` statement:

```
automatic-site-id {
    collision-detect-time seconds;
    new-site-wait-time seconds;
    reclaim-wait-time minimum seconds maximum seconds;
    startup-wait-time seconds;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

The `automatic-site-id` statement includes a number of options that control different delays in network layer reachability information (NLRI) advertisements. All of these options are configured with default values. See the statement summary for the `automatic-site-id` statement for more information.

The `automatic-site-id` statement includes the following options:

- `collision-detect-time`—The time in seconds to wait after a claim advertisement is sent to the other routers in a VPLS instance before a PE router can begin using a site identifier. If the PE router receives a competing claim advertisement for the same site identifier during this time period, it initiates the collision resolution procedure for site identifiers.
- `new-site-wait-time`—The time in seconds to wait to receive VPLS information for a newly configured routing instance or a new site. This time interval is also applied whenever the automatic site identifier feature is activated on a VPLS routing instance other than at startup. Effectively, this timer indicates how long to wait before an attempt is made to allocate a site identifier. This timer is also triggered whenever a VPLS routing instance is enabled.
- `reclaim-wait-time`—The time to wait before attempting to claim a site identifier after a collision. A collision occurs whenever an attempt is made to claim a site identifier by two separate VPLS sites.

- `startup-wait-time`—The time in seconds to wait at startup to receive all the VPLS information for the route targets configured on the other PE routers included in the VPLS routing instance.

Configuring the Site Range

When you enable BGP signaling for each VPLS routing instance, you can optionally configure the site range. The site range specifies an upper limit on the maximum site identifier that can be accepted to allow a pseudowire to be brought up. You must specify a value from 1 through 65,534. The default value is 65,534. We recommend using the default. Pseudowires cannot be established to sites with site identifiers greater than the configured site range. If you issue the `show vpls connections` command, such sites are displayed as OR (out of range).

To configure the site range, include the `site-range` statement:

```
site-range number;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols vpls]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

There are networks that require that the site range be configured using a value smaller than the local site identifier, for example, a hub-and-spoke VPLS with multihomed sites. For this type of network, you need to allow pseudowires to be established between the spoke routers and the hub router. However, you also need to prevent pseudowires from being established between spoke routers directly. Due to the multihoming requirement of spoke sites, Layer 2 VPN NRLIs need to be accepted from other spoke routers (at least from spokes with the same site identifier as the locally configured sites) to determine the status of local spoke routers (active or not active) based on the local preference included in the NRLIs received from the other spoke routers.

This type of VPLS network can be implemented by, for example, numbering hub sites with identifiers 1 through 8 and spoke sites with identifiers 9 and larger. You can then configure a site range of 8 on each of the spoke sites. Although the spoke sites accept NRLIs and install them in the Layer 2 VPN routing tables (allowing the multihomed sites to determine the status of the local site), the spoke sites cannot establish pseudowires directly to the other spoke sites due to the configured site range.

The following configurations illustrate this concept. The configurations are for the VPLS routing instances on three routers, two spoke routers and one hub router:

Router 1—spoke:

```
routing-instance hub-and-spoke {
    no-local-switching;
    protocols {
        vpls {
            site-range 8;
            no-tunnel-services;
            site spoke-9 {
                site-identifier 9 {
                    multi-homing;
                    site-preference primary;
                }
            }
            site spoke-10 {
                site-identifier 10 {
                    multi-homing;
                    site-preference backup;
                }
            }
        }
    }
}
```

Router 2—spoke:

```
routing-instance hub-and-spoke {
    no-local-switching;
    protocols {
        vpls {
            site-range 8;
            no-tunnel-services;
            site spoke-9 {
                site-identifier 9 {
                    multi-homing;
                    site-preference backup;
                }
            }
            site spoke-10 {
                site-identifier 10 {
                    multi-homing;
                    site-preference primary;
                }
            }
        }
    }
}
```

Hub—router 3:

```
routing-instance hub-and-spoke {
    no-local-switching;
    protocols {
        vpls {
            no-tunnel-services;
            site hub {
                site-identifier 1;
            }
        }
    }
}
```

Configuring the VPLS Site Interfaces

You must configure an interface for each of the pseudowires you specify for the VPLS site.

To configure an interface for the VPLS site, include the `interface` statement:

```
interface interface-name {
    interface-mac-limit limit;
}
```

You can include these statements at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

You can also configure a limit on the number of MAC addresses that can be learned from the specified interface. For more information, see "Limiting the Number of MAC Addresses Learned from an Interface" on page 690.

Configuring the VPLS Site Preference

You can specify the local preference value advertised for a particular VPLS site. The site preference value is specified using the `site-preference` statement configured at the `[edit routing-instances *routing-instance-name* protocols vpls site *site-name*]` hierarchy level. By configuring the `site-preference` statement, a value configured for the `local-preference` statement at the `[edit protocols bgp]` hierarchy level is ignored by the VPLS routing instance. However, you can change the site preference value for VPLS routes exported to other routers by configuring an export policy. When a PE router receives multiple advertisements with the same VPLS edge (VE) device identifier, the advertisement with the highest local preference value is preferred.

To configure the VPLS site preference, include the `site-preference` statement:

```
site-preference preference-value {
    backup;
    primary;
}
```

You can also specify either the `backup` option or the `primary` option for the `site-preference` statement. The `backup` option specifies the preference value as 1, the lowest possible value, ensuring that the VPLS site is the least likely to be selected. The `primary` option specifies the preference value as 65,535, the highest possible value, ensuring that the VPLS site is the most likely to be selected.

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

Configuring LDP Signaling for VPLS

You can configure LDP as the signaling protocol for a VPLS routing instance. This functionality is described in RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*.

The Junos OS software does not support all of RFC 4762. When enabling LDP signaling for a VPLS routing instance, network engineers should be aware that only the following values are supported:

- FEC—`128` or `129`
- Control bit—`0`
- Ethernet pseudowire type—`0x0005`
- Ethernet tagged mode pseudowire type—`0x0004`

LDP signaled VPLS supports the Virtual Circuit Connectivity Verification (VCCV) Type Length Value (TLV) for pseudowire label mapping, label database display, and LDP trace. When you enable LDP signaling for a pseudowire, LDP advertises the VCCV capabilities to the neighboring routers. VCCV provides a control channel for a pseudowire and includes both operations and management functions (for example, connectivity verification). This control channel is established between the pseudowire's ingress and egress devices. Once established, connectivity verification messages can be sent over the VCCV control channel.

The Junos OS software supports the following VCCV capabilities for LDP signaled VPLS (defined in RFC 5085 Section 8.1):

- VCCV connectivity check types:
  - Router Alert Label
  - MPLS pseudowire label with TTL=1
- VCCV connectivity verification type:
  - LSP ping

If the peer device also advertises VCCV parameters during pseudowire setup, the Junos OS software selects the set of common advertised parameters to use as the method for performing VCCV OAM on the pseudowire.

The locally advertised and peer advertised VCCV parameters can be viewed using the `show ldp database` command as show here:

```
user@host> show ldp database l2circuit extensive
Input label database, 10.255.245.198:0--10.255.245.194:0
  Label     Prefix
 299872     L2CKT CtrlWord PPP VC 100
            MTU: 4470
            VCCV Control Channel types:
                MPLS router alert label
                MPLS PW label with TTL=1
            VCCV Control Verification types:
                LSP ping
  Label     Prefix
            State: Active
            Age: 19:23:08
```

Be aware of the following behavior with regard to TLVs when configuring LDP-signaled VPLS in a network with equipment from other vendors:

- When a Juniper Network's device receives a TLV with an empty address, LDP accepts the TLV.
- When a MAC address is withdrawn, LDP specifies a zero address (0.0.0.0) for the AddressList.

To enable LDP signaling for the set of PE routers participating in the same VPLS routing instance, you need to use the `vpls-id` statement configured at the `[edit routing-instances *routing-instance-name* protocols vpls]` hierarchy level to configure the same VPLS identifier on each of the PE routers. The VPLS identifier must be globally unique. When each VPLS routing instance (domain) has a unique VPLS identifier, it is possible to configure multiple VPLS routing instances between a given pair of PE routers.

LDP signaling requires that you configure a full-mesh LDP session between the PE routers in the same VPLS routing instance. Neighboring PE routers are statically configured. Tunnels are created between the neighboring PE routers to aggregate traffic from one PE router to another. Pseudowires are then signaled to demultiplex traffic between VPLS routing instances. These PE routers exchange the pseudowire label, the MPLS label that acts as the VPLS pseudowire demultiplexer field, by using LDP forwarding equivalence classes (FECs). Tunnels based on both MPLS and generic routing encapsulation (GRE) are supported.

NOTE: You cannot configure both BGP signaling and LDP signaling for the same VPLS routing instance. If you attempt to configure the statements that enable BGP signaling for the VPLS routing instance (the `site`, `site-identifier`, and `site-range` statements), and the statements that enable LDP signaling for the same instance, `neighbor` and `vpls-id`, the commit operation fails.

To enable LDP signaling for the VPLS routing instance, complete the steps in the following sections:

Configuring LDP Signaling for the VPLS Routing Instance

To configure the VPLS routing instance to use LDP signaling, you must configure the same VPLS identifier on each PE router participating in the instance. Specify the VPLS identifier with the `vpls-id` statement:

```
vpls-id vpls-id;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances *routing-instance-name* protocols vpls]`
- `[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

To configure the VPLS routing instance to use LDP signaling, you also must include the `neighbor` statement to specify each of the neighboring PE routers that are a part of this VPLS domain:

```
neighbor neighbor-id;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

Configuring LDP Signaling on the Router

To enable LDP signaling, you need to configure LDP on each PE router participating in the VPLS routing instance. A minimal configuration is to enable LDP on the loopback interface, which includes the router identifier (`router-id`), on the PE router using the `interface` statement:

```
interface interface-name;
```

You can include this statement at the following hierarchy levels:

- `[edit protocols ldp]`
- `[edit logical-systems` *`logical-system-name`* `protocols ldp]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

You can enable LDP on all the interfaces on the router using the `all` option for the `interfaces` statement. For more information about how to configure LDP, see the MPLS Applications User Guide.

Configuring VPLS Routing Instance and VPLS Interface Connectivity

You can configure the VPLS routing instance to take down or maintain its VPLS connections depending on the status of the interfaces configured for the VPLS routing instance. By default, the VPLS connection is taken down whenever a customer-facing interface configured for the VPLS routing instance fails. This behavior can be explicitly configured by specifying the `ce` option for the `connectivity-type` statement:

```
connectivity-type ce;
```

You can alternatively specify that the VPLS connection remain up so long as an Integrated Routing and Bridging (IRB) interface is configured for the VPLS routing instance by specifying the `irb` option for the `connectivity-type` statement:

```
connectivity-type irb;
```

To ensure that the VPLS connection remain up until explicitly taken down, specify the `permanent` option for the `connectivity-type` statement:

```
connectivity-type permanent;
```

This option is reserved for use in configuring Layer 2 Wholesale subscriber networks. See the *Broadband Subscriber Management Solutions Guide* for details about configuring a Layer 2 Wholesale network.

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.
ACX Series routers do not support irb interface in VPLS instance, therefore connectivity-type irb for VPLS is not supported.

Configuring the VPLS Encapsulation Type

You can specify a VPLS encapsulation type for the pseudowires established between VPLS neighbors. The encapsulation type is carried in the LDP-signaling messages exchanged between VPLS neighbors when pseudowires are created. You might need to alter the encapsulation type depending on what other vendors' equipment is deployed within your network.

VPLS effectively provides a bridge between Ethernet networks. As a consequence, only two encapsulation types are available:

- `ethernet`—Ethernet

- `ethernet-vlan`—Ethernet virtual LAN (VLAN)

If you do not specify an encapsulation type for the VPLS routing instance or the VPLS neighbor, `ethernet` is used.

To specify an encapsulation type for the VPLS routing instance, include the `encapsulation-type` statement:

```
encapsulation-type (ethernet | ethernet-vlan);
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

You can also specify an encapsulation type for a specific VPLS neighbor by including the `encapsulation-type` statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`*`]`

Configuring the MPLS Routing Table to Leak Routes a Nondefault Routing Instance

Starting in Junos OS Release 16.1, you can specify one or more nondefault routing instances where you want MPLS routes to be leaked from the mpls.0 path routing table in the primary routing instance. This capability is useful in an L2VPN/VPLS configuration when the remote PE router is learned from the IGP in a nondefault routing instance, because L2VPN/VPLS installs ingress-labeled routes only in the primary mpls.0 table.

By default, routes in the mpls.0 routing table in the primary routing instance are not leaked to the corresponding routing tables in nondefault routing instances. When L2VPN/VPLS traffic is received on the core-facing interface in a nondefault routing instance, the router performs a lookup in the table that corresponds to that interface, *routing-instance-name*.mpls.0. Because the routes are not leaked by default, then no routes are found in the *routing-instance-name*.mpls.0 routing table and all the incoming traffic is dropped.

To leak MPLS routes to a nondefault routing instance, include the `import-labeled-routes` statement and specify one or more routing instances where the routes need to be leaked:

```
import-labeled-routes [ routing-instance-name ];
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

Configuring the VPLS MAC Table Timeout Interval

You can modify the timeout interval for the VPLS table. We recommend you that configure longer values for small, stable VPLS networks and shorter values for large, dynamic VPLS networks. If the VPLS table does not receive any updates during the timeout interval, the router waits one additional interval before automatically clearing the MAC address entries from the VPLS table.

To modify the timeout interval for the VPLS table, include the `mac-table-aging-time` statement:

```
mac-table-aging-time seconds;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

NOTE: The `mac-table-aging-time` statement is not available on ACX Series and MX Series routers.

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

Configuring the Size of the VPLS MAC Address Table

You can modify the size of the VPLS media access control (MAC) address table. The default table size is 512 MAC addresses, the minimum is 16 addresses, and the maximum is 65,536 addresses.

NOTE: T4000 routers with Type 5 FPCs support up to 262,143 MAC addresses per VPLS routing instance. To enable the improved VPLS MAC address learning limit (that is, 262,143 MAC addresses), you must Include the `enhanced-mode` statement at the `[edit chassis network-services]` hierarchy level, reboot the router, and then modify the size of the VPLS MAC address table.

If the MAC table limit is reached, new MAC addresses can no longer be added to the table. Eventually the oldest MAC addresses are removed from the MAC address table automatically. This frees space in the table, allowing new entries to be added. However, as long as the table is full, new MAC addresses are dropped.

To change the VPLS MAC table size for each VPLS or VPN routing instance, include the `mac-table-size` statement:

```
mac-table-size size;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name protocols vpls]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

When you include the `mac-table-size` statement, the affected interfaces include all interfaces within the VPLS routing instance, including the local interfaces, the LSI interfaces, and the VT interfaces.

NOTE: ACX Series routers do no support `mac-table-size` statement for VPLS.

Limiting the Number of MAC Addresses Learned from an Interface

You can configure a limit on the number of MAC addresses learned by a VPLS routing instance using the `mac-table-size` statement. If the MAC table limit is reached, new MAC addresses can no longer be added to the table. Eventually the oldest MAC addresses are removed from the MAC address table automatically. This frees space in the table, allowing new entries to be added. However, as long as the table is full, new MAC addresses are dropped.

Because this limit applies to each VPLS routing instance, the MAC addresses of a single interface can consume all the available space in the table, preventing the routing instance from acquiring addresses from other interfaces.

You can limit the number of MAC addresses learned from each interface configured for a VPLS routing instance. To do so, include the `interface-mac-limit` statement:

```
interface-mac-limit limit;
```

NOTE: ACX Series routers do not support *interface-mac-limit limit* for VPLS.

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

The `interface-mac-limit` statement affects the local interfaces only (the interfaces facing CE devices).

Configuring the `interface-mac-limit` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level causes the same limit to be applied to all of the interfaces configured for that specific routing instance.

NOTE: Starting in Junos OS Release 12.3R4, if you do not configure the parameter to limit the number of MAC addresses to be learned by a VPLS instance, the default value is not effective. Instead, if you do not include the `interface-mac-limit` option at the `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`* `interfaces` *`interface-name`*`]`, hierarchy level, this setting is not present in the configuration with the default value of 1024 addresses. If you upgrade a router running a Junos OS release earlier than Release 12.3R4 to Release 12.3R4 or later, you must configure the `interface-mac-limit` option with a valid value for it to be saved in the configuration.

You can also limit the number of MAC addresses learned by a specific interface configured for a VPLS routing instance. This gives you the ability to limit particular interfaces that you expect might generate a lot of MAC addresses.

To limit the number of MAC addresses learned by a specific interface, include the `interface-mac-limit` statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`* `interfaces` *`interface-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`* `interfaces` *`interface-name`*`]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

The MAC limit configured for an individual interface at this hierarchy level overrides any value configured at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level. Also, the MAC limit configured using the `mac-table-size` statement can override the limit configured using the `interface-mac-limit` statement.

The MAC address limit applies to customer-facing interfaces only.

Removing Addresses from the MAC Address Database

You can enable MAC flush processing for the VPLS routing instance or for the mesh group under a VPLS routing instance. MAC flush processing removes MAC addresses from the MAC address database that have been learned dynamically. With the dynamically learned MAC addresses removed, MAC address convergence requires less time to complete.

You can clear dynamically learned MAC addresses from the MAC address database by including the `mac-flush` statement:

```
mac-flush [ explicit-mac-flush-message-options ];
```

To clear dynamically learned MAC addresses globally across all devices participating in the routing instance, you can include the statement at the following hierarchy levels:

- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`

To clear the MAC addresses on the routers in a specific mesh group, you can include the statement at the following hierarchy levels:

- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`*`]`
- `[edit routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`*`]`

NOTE: On ACX Series routers, the `mesh-group` statement is supported only on ACX5000 line of routers. ACX5000 line of routers can support up to 8 user-defined mesh groups per VPLS routing instance.

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

For certain cases where MAC flush processing is not initiated by default, you can also specify *`explicit-mac-flush-message-options`* to additionally configure the router to send explicit MAC flush messages under specific conditions. For a list of the explicit MAC flush message options you can include with this statement, see the summary section for this statement.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
16.1	Starting in Junos OS Release 16.1, you can specify one or more nondefault routing instances where you want MPLS routes to be leaked from the mpls.0 path routing table in the primary routing instance.
12.3R4	Starting in Junos OS Release 12.3R4, if you do not configure the parameter to limit the number of MAC addresses to be learned by a VPLS instance, the default value is not effective.

RELATED DOCUMENTATION

Configuring Improved VPLS MAC Address Learning on T4000 Routers with Type 5 FPCs | **1225**

enhanced-mode

show ldp database

Configuring a VPLS Routing Instance

Use the **vpls** routing instance type for point-to-multipoint LAN implementations between a set of sites in a VPN.

To create a routing instance for VPLS, include at least the following statements in the configuration:

```
routing-instances {
    routing-instance-name {
        instance-type vpls;
        interface interface-name;
        route-distinguisher (as-number:number | ip-address:number);
        vrf-import [ policy-names ];
        vrf-export [ policy-names ];
        protocols {
            vpls {
                ... vpls configuration ...
            }
        }
    }
}
```

You can include these statements at the following hierarchy levels:

- `[edit]`
- `[edit logical-systems` *`logical-system-name`*`]`

For more information about configuring VPLS, see theJunos OS VPNs Library for Routing Devices.

RELATED DOCUMENTATION

Layer 2 Routing Instance Types

Configuring a Virtual Switch Routing Instance on MX Series Routers

Configuring a Layer 2 Control Protocol Routing Instance | **698**

Support of Inner VLAN List and Inner VLAN Range for Qualified BUM Pruning on a Dual-Tagged Interface for a VPLS Routing Instance Overview

Junos OS provides the `qualified-bum-pruning-mode` statement, which supports constraining of broadcast, multicast, and unknown (BUM) traffic in a VPLS instance to a specific subscriber VLAN stack. This cuts down unnecessary consumption of bandwidth and thereby, improves network performance.

All subscriber VLANs that need to be backhauled to their respective retail-ISPs are also created in the same VPLS instance as the retail-ISP. Each retail-ISP is allocated a VPLS instance. Any traffic on subscriber VLANs is hauled over the VPLS tunnel to the retail-ISP. Similarly, any traffic from a retail-ISP on a VPLS tunnel is forwarded to all the subscriber VLANs in that VPLS instance. For multicast traffic (which includes broadcast, unknown DMAC, and layer 2 and layer 3 multicast), the standard Layer-2 VPLS instance-forwarding creates a BUM packet, which comes from a retail-ISP over a VPLS tunnel, to be flooded to all subscriber VLANs in the VPLS instance. This causes the individual subscribers who might have subscribed to specific multicast traffic channels to receive all the traffic instead of the multicast traffic that the subscriber has signed up for.

To overcome this, BUM traffic needs to be forwarded only to subscribers who are the intended recipients, by mapping the VLAN-tags present in the BUM packet to these subscribers. A subscriber is assigned a stacked VLAN-tag and BUM packets are sent only to a subscriber whose stacked VLAN-tags match the VLAN-tags present in the BUM packet. This ensures that subscribers receive only the BUM traffic that is specifically intended for them, which prevents the normal flooding of BUM traffic. This is called *BUM-pruning*. Hitherto, Junos OS supported only single VLAN tagged and dual VLAN tagged subscriber interfaces. This implies that for each subscriber, a different interface must be configured. Such a BUM-pruning solution does not scale well. To address this issue, Junos OS now supports configuring VLAN ranges on the subscriber interfaces. This enables better management of subscriber services.

BUM-pruning on Junos OS allows subscriber interfaces to be configured with inner-vlan-lists. Each inner VLAN list includes all the subscriber VLANs that must be grouped on a particular subscriber interface mapped to a VPLS instance. You can configure BUM-pruning on each VPLS instance allocated to a retail-ISP by using the `set routing-instances` *`routing-instance-name`* `qualified-bum-pruning-mode` command.

For normal VPLS flooding, BUM traffic is received over the VPLS tunnel and flooded to all the subscriber interfaces mapped to the vpls-instance. These interfaces might be or might not be mapped to the same subscriber VLAN as the packet received over the VPLS tunnel. By enabling the qualified BUM pruning mode in a VPLS instance, VPLS flooding is restricted to a combination of service-provider VLAN and subscriber VLAN. The `qualified-bum-pruning` statement implements BUM-pruning on the Packet Forwarding Engine in the egress list of each subscriber interface mapped to the VPLS instance. On the Packet Forwarding Engine, BUM-pruning is implemented as a vlan-check nexthop installed in the egress list of the subscriber interface. The vlan-check nexthop checks whether the BUM packet exiting the ifl has the same combination of service-provider VLAN and subscriber VLAN as that configured on the interface. If the VLAN check matches, the packet is forwarded or else it is discarded. This ensures that only the subscriber that is the intended recipient of the BUM traffic receives the packet. This feature is supported for both single-tagged and dual-tagged subscriber interfaces as well as for subscriber interfaces configured with vlan-map operations. If vlan-map operations are configured on the interfaces, then the normalized VLAN on the interface is considered for the vlan-check nexthop. This feature is supported for both default and logical systems.

Currently, dual-tagged subscriber interfaces can support a single pair of service-provider VLAN and subscriber VLAN. With the support for BUM-pruning of VPLS traffic on dual-tagged interfaces, you can configure a single service provider VLAN(S) on the subscriber interface and map it to multiple customer VLANs using a single inner VLAN list or inner VLAN range. The inner VLAN list on a subscriber interface can have multiple elements. Each element of the inner VLAN list can be as follows:

- A single VLAN tag
- A range of VLANs

The BUM traffic flow through dual-tagged interfaces is supported on both aggregated and non-aggregated subscriber interfaces. When BUM traffic exits a subscriber interface configured with inner VLAN list or inner VLAN range, the service provider VLAN (S) and the subscriber VLAN (C) in the packet are checked against all combinations of (S,C) of the S and C VLANs possible on the interface. If the packet matches any of the combination, it is forwarded on the subscriber interface. If the packet does not match any combination, it is discarded. If the subscriber has vlan-map configured, then the S and C VLANs to be checked are modified based on the VLAN normalization on the interface.

RELATED DOCUMENTATION

qualified-bum-pruning-mode

vlan-tags

Configuring Qualified BUM Pruning for a Dual-Tagged Interface with Inner VLAN list and InnerVLAN range for a VPLS Routing Instance

Currently, dual-tagged subscriber interfaces can support a single pair of service-provider VLAN and subscriber VLAN. With the support for broadcast, unknown unicast, multicast (BUM) pruning of VPLS traffic on dual-tagged interfaces, you can configure a single service provider VLAN— referred to as VLAN(S) in this topic —on the subscriber interface and map it to multiple customer VLANs by using a single inner VLAN list or inner VLAN range. The inner VLAN list on a subscriber interface can have multiple elements. Each element of the inner VLAN list can be a single VLAN tag or a range of VLANs. The BUM traffic flow through dual-tagged interfaces is supported on both aggregated and nonaggregated subscriber interfaces. When BUM traffic exits a subscriber interface that has an inner VLAN list or range configured, the VLAN(S) and the subscriber VLAN— referred to as VLAN(C)— in the packet are checked against all combinations of VLAN(S) and VLAN(C) possible on the interface. If the packet contents match any of the combinations, then the packet is forwarded on the subscriber interface. If the contents do not match any combination, then the packet is discarded. If the subscriber has a VLAN map configured, then the VLAN(S) and VLAN(C) to be checked are modified based on the VLAN normalization on the interface.

Before you configure qualified BUM pruning for a dual-tagged interface including inner VLAN list and inner VLAN range for a VPLS routing instance, you must do the following:

1. Configure the device interfaces.

2. Configure the VPLS routing instance.

To configure qualified-bum-pruning for dual tagged interface including inner VLAN list and inner VLAN range for a VPLS routing instance:

1. Configure a member interface with a service provider VLAN for the VPLS routing instance. You can configure service provider VLAN with an inner VLAN list or inner VLAN range for the VPLS routing instance.

   ```
   [edit interfaces interface-name unit unit-number vlan-tags]
   user@host# set outer vlan-id
   ```

For example, to configure member interface with the service provider VLAN ID 200 for a VPLS routing instance:

```
[edit interfaces ae0 unit 100 vlan-tags]
user@host# set outer 200
```

2. Configure the `inner-list` of the member interface with a single customer VLAN ID or a range of customer VLAN IDs or both for a VPLS routing instance.

```
[edit interfaces interface-name unit unit-number vlan-tags]
user@host# set inner-list [vlan-id]
```

For example, to configure the member interface with a single customer VLAN ID and a range of customer VLAN IDs for a VPLS routing instance:

```
[edit interfaces ae0 unit 100 vlan-tags]
user@host# set inner-list [210 -215 216]
```

3. Configure the member interface with inner VLAN ID range.

```
[edit interfaces interface-name unit unit-number vlan-tags]
user@host# set inner-range vlan-range
```

For example, to configure the member interface with an inner range of 300-310 for VPLS routing instance:

```
[edit interfaces ae6 unit 500 vlan-tags]
user@host# set inner-range 300-310
```

4. Configure BUM pruning for VPLS traffic on dual-tagged interfaces to forward the BUM traffic only to the intended member interfaces.

```
[edit routing-instance VPLS routing-instance]
user@host# set qualified-bum-pruning-mode
```

For example, to configure qualified BUM pruning for routing instance r1:

```
[edit routing-instance r1]
user@host# set qualified-bum-pruning-mode
```

RELATED DOCUMENTATION

Support of Inner VLAN List and Inner VLAN Range for Qualified BUM Pruning on a Dual-Tagged Interface for a VPLS Routing Instance Overview | **694**

qualified-bum-pruning-mode

vlan-tags

Configuring a Layer 2 Control Protocol Routing Instance

On MX Series routers only, use the **layer2-control** routing instance type for Rapid Spanning-Tree Protocol (RSTP) or Multiple Spanning-Tree Protocol (MSTP) in customer edge interfaces of a VPLS routing instance. Layer 2 control protocols enable features such as Layer 2 protocol tunneling or nonstop bridging. This instance type cannot be used if the customer edge interface is multihomed to two provider edge interfaces. If the customer edge interface is multihomed to two provider edge interfaces, use the default bridge protocol data unit (BPDU) tunneling.

To create a routing instance for Layer 2 control protocols, include at least the following statements in the configuration:

```
routing-instances {
    routing-instance-name {
        instance-type layer2-control;
        interface interface-name;
        route-distinguisher (as-number:number | ip-address:number);
        vrf-import [ policy-names ];
        vrf-export [ policy-names ];
        protocols {
            mstp {
                ... interface options ...
                msti msti-id {
                    ... MSTP MSTI configuration ...
                }
            }
        }
    }
}
```

You can include these statements at the following hierarchy levels:

- `[edit]`
- `[edit logical-systems` *`logical-system-name`*`]`

RELATED DOCUMENTATION

Layer 2 Routing Instance Types

Configuring a VPLS Routing Instance | **693**

Configuring a Virtual Switch Routing Instance on MX Series Routers

Layer 2 Protocol Tunneling (L2PT)

Configuring RSTP

Configuring MSTP

PE Router Mesh Groups for VPLS Routing Instances

A PE router mesh group consists of a set of routers participating in a VPLS routing instance that share the same signaling protocol, either BGP or LDP. Each VPLS routing instance can have just one BGP mesh group. However, you can configure multiple LDP mesh groups for each routing instance.

The Junos OS can support up to 16 mesh groups on MX Series routers. However, two mesh groups are created by default, one for the CE routers and one for the PE routers. Therefore, the maximum number of user-defined mesh groups is 14 for MX Series routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The Junos OS supports both forwarding equivalency class (FEC) 128 and FEC 129. FEC 129 uses VPLS autodiscovery to convey endpoint information. FEC 128 requires manually configured pseudowires.

The following describes the behavior of mesh groups in regards to BGP-signaled PE routers and LDP-signaled PE routers:

- BGP-signaled PE routers—Automatically discovered PE routers that use BGP for signaling are associated with the default VE mesh group. You cannot configure the Junos OS to associate these routers with a user-defined VE mesh group.
- LDP-signaled PE routers (FEC 128)—PE routers statically configured using FEC-128 LDP signaling are placed in a default mesh group. However, you can configure a VE mesh group and associate each LDP FEC-128 neighbor with it. Each configured VE mesh group contains a set of VEs that are in the same interior gateway protocol (IGP) routing instance and are fully meshed with each other in the control and data planes.
- LDP-signaled PE routers (FEC 129)—Configuration for a mesh group for FEC 129 is very similar to the configuration for FEC 128.

  Note the following differences for FEC 129:

  - Each user-defined mesh group must have a unique route distinguisher. Do not use the route distinguisher that is defined for the default mesh group at the `[edit routing-intances]` hierarchy level.
  - Each user-defined mesh group must have its own import and export route target.
  - Each user-defined mesh group can have a unique Layer 2 VPN ID. By default, all the mesh groups that are configured for the a VPLS routing-instance use the same Layer 2 VPN ID, the one that you configure at the `[edit routing-instances]` hierarchy level.

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | **913**

Configuring VPLS Fast Reroute Priority

When a path is rerouted after a link failure by using the MPLS fast reroute feature, the router repairs the affected next hops by switching them from the active label switched path (LSP) to the standby LSP. To specify the order in which the router repairs next hops and restores traffic convergence for VPLS routing instances after a fast reroute event, you can use the `fast-reroute-priority` statement to configure `high`, `medium`, or `low` fast reroute priority for a VPLS routing instance. By default, the fast reroute priority for a VPLS routing instance is `low`.

The router repairs next hops and restores known unicast, unknown unicast, broadcast, and multicast traffic for VPLS routing instances in the following order, based on the fast reroute priority configuration:

1. The router repairs next hops for high-priority VPLS routing instances.

2. The router repairs next hops for medium-priority VPLS routing instances.

3. The router repairs next hops for low-priority VPLS routing instances.

Because the router repairs next hops for VPLS routing instances configured with `high` fast reroute priority first, the traffic traversing high-priority VPLS instances is restored faster than the traffic for VPLS instances configured with `medium` or `low` fast reroute priority. The ability to prioritize specific VPLS routing instances for faster convergence and traffic restoration enables service providers to offer differentiated service levels to their customers.

Within a particular fast reroute priority level (`high`, `medium`, or `low`), the router follows no particular order for traffic restoration of VPLS routing instances.

NOTE: VPLS fast reroute priority is not supported on EX Series switches.

To configure `high`, `medium`, or `low` fast reroute priority for a VPLS routing instance, include the `fast-reroute-priority` statement:

```
fast-reroute-priority (high | medium | low);
```

You can include this statement at the following hierarchy levels:

- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `forwarding-options]`
- `[edit routing-instances` *`routing-instance-name`* `forwarding-options]`

You can configure fast reroute priority only for routing instances with the `instance-type` set to `vpls`. If you attempt to configure fast reroute priority for a routing instance with an `instance-type` other than `vpls`, the router displays a warning message and the configuration fails.

The following example snippet shows configuration of `high` fast reroute priority for a VPLS routing instance named `test-vpls`:

```
test-vpls {
    instance-type vpls;
    forwarding-options {
        fast-reroute-priority high;
    }
}
```

To display the fast reroute priority setting configured for a VPLS routing instance, use the `show route instance detail` operational command. For information about using this command, see the CLI Explorer.

Specifying the VT Interfaces Used by VPLS Routing Instances

By default, the Junos OS automatically selects one of the virtual tunnel (VT) interfaces available to the router for de-encapsulating traffic from a remote site. The Junos OS cycles through the currently available VT interfaces, regularly updating the list of available VT interfaces as new remote sites are discovered and new connections are brought up. However, you can also explicitly configure which VT interfaces will receive the VPLS traffic.

By including the `tunnel-services` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level, you can specify that traffic for particular VPLS routing instances be forwarded to specific VT interfaces. Doing so allows you to load-balance VPLS traffic among all the available VT interfaces on the router.

The `tunnel-services` statement includes the following options:

- `devices`—Specifies the VT interfaces acceptable for use by the VPLS routing instance. If you do not configure this option, all VT interfaces available to the router can be used for de-encapsulating traffic for this instance.
- `primary`—Specifies the primary VT interface to be used by the VPLS routing instance. The VT interface specified is used to de-encapsulate all VPLS traffic from the MPLS core network for this routing instance. If the VT interface specified is unavailable, then one of the other acceptable VT interfaces (specified in the devices option) is used for handling the VPLS traffic. If you do not configure this option, any acceptable VT interface can be used to de-encapsulate VPLS traffic from the core.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

To specify that traffic for a particular VPLS routing instance be forwarded to specific VT interfaces, include the `tunnel-services` statement:

```
tunnel-services {
    devices device-names;
    primary primary-device-name;
}
```

These statements can be configured at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

Understanding PIM Snooping for VPLS

There are two ways to direct PIM control packets:

- By the use of PIM snooping
- By the use of PIM proxying

PIM snooping configures a device to examine and operate only on PIM hello and join/prune packets. A PIM snooping device snoops PIM hello and join/prune packets on each interface to find interested multicast receivers and populates the multicast forwarding tree with this information. PIM snooping differs from PIM proxying in that both PIM hello and join/prune packets are transparently flooded in the VPLS as opposed to the flooding of only hello packets in the case of PIM proxying. PIM snooping is configured on PE routers connected through pseudowires. PIM snooping ensures that no new PIM packets are generated in the VPLS, with the exception of PIM messages sent through LDP on pseudowires.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

A device that supports PIM snooping snoops hello packets received on attachment circuits. It does not introduce latency in the VPLS core when it forwards PIM join/prune packets.

To configure PIM snooping on a PE router, use the `pim-snooping` statement at the `[edit routing-instances` *`instance-name`* `protocols]` hierarchy level:

```
routing-instances {
    customer {
        instance-type vpls;
        ...
        protocols {
            pim-snooping{
                traceoptions {
                    file pim.log size 10m;
                    flag all;
                    flag timer disable;
                }
            }
        }
    }
}
```

Example: Configuring PIM Snooping for VPLS explains the PIM snooping method. The use of the PIM proxying method is not discussed here and is outside the scope of this document. For more information about PIM proxying, see PIM Snooping over VPLS.

Example: Configuring PIM Snooping for VPLS

IN THIS SECTION

- Requirements | 704
- Overview | 704
- Configuration | 706
- Verification | 715

This example shows how to configure PIM snooping in a virtual private LAN service (VPLS) to restrict multicast traffic to interested devices.

Requirements

This example uses the following hardware and software components:

- M Series Multiservice Edge Routers (M7i and M10i with Enhanced CFEB, M120, and M320 with E3 FPCs) or MX Series 5G Universal Routing Platforms (MX80, MX240, MX480, and MX960)
- Junos OS Release 13.2 or later

Overview

IN THIS SECTION

- Topology | 705

The following example shows how to configure PIM snooping to restrict multicast traffic to interested devices in a VPLS.

NOTE: This example demonstrates PIM snooping by the use of a PIM snooping device to restrict multicast traffic. The use of the PIM proxying method to achieve PIM snooping is out of the scope of this document and is yet to be implemented in Junos OS.

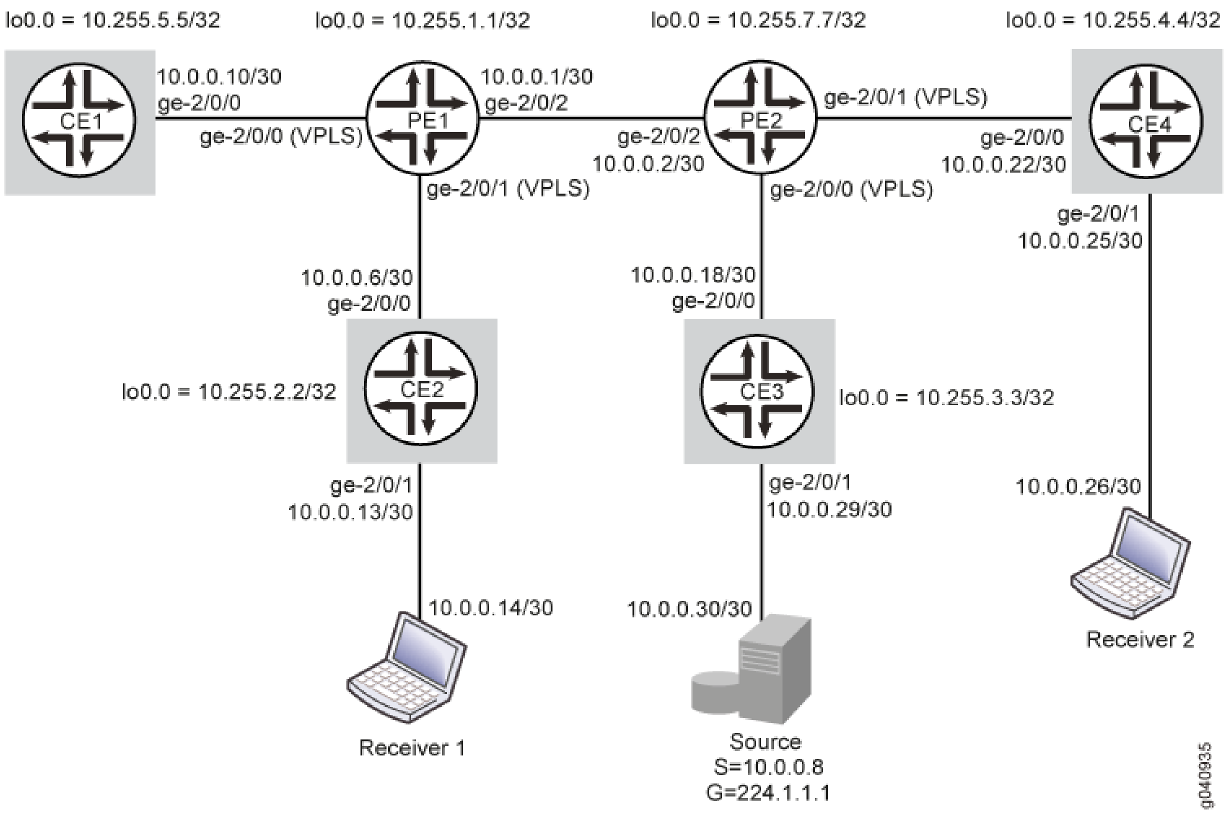
Topology

In this example, two PE routers are connected to each other through a pseudowire connection. Router PE1 is connected to Routers CE1 and CE2. A multicast receiver is attached to Router CE2. Router PE2 is connected to Routers CE3 and CE4. A multicast source is connected to Router CE3, and a second multicast receiver is attached to Router CE4.

PIM snooping is configured on Routers PE1 and PE2. Hence, data sent from the multicast source is received only by members of the multicast group.

Figure 53 on page 705 shows the topology used in this example.

Figure 53: PIM Snooping for VPLS



Configuration

IN THIS SECTION

- CLI Quick Configuration | 706
- Configuring PIM Snooping for VPLS | 709
- Results | 712

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 PE1

```
set multicast-snooping-options traceoptions file snoop.log size 10m
set interfaces ge-2/0/0 encapsulation ethernet-vpls
set interfaces ge-2/0/0 unit 0 description toCE1
set interfaces ge-2/0/1 encapsulation ethernet-vpls
set interfaces ge-2/0/1 unit 0 description toCE2
set interfaces ge-2/0/2 unit 0 description toPE2
set interfaces ge-2/0/2 unit 0 family inet address 10.0.0.1/30
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.1.1/32
set routing-options router-id 10.255.1.1
set protocols mpls interface ge-2/0/1.0
set protocols bgp group toPE2 type internal
set protocols bgp group toPE2 local-address 10.255.1.1
set protocols bgp group toPE2 family l2vpn signaling
set protocols bgp group toPE2 neighbor 10.255.7.7
set protocols ospf area 0.0.0.0 interface ge-2/0/2.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-2/0/2.0
set protocols ldp interface lo0.0
set routing-instances titanium instance-type vpls
set routing-instances titanium vlan-id none
set routing-instances titanium interface ge-2/0/0.0
set routing-instances titanium interface ge-2/0/1.0
set routing-instances titanium route-distinguisher 101:101
set routing-instances titanium vrf-target target:201:201
set routing-instances titanium protocols vpls vpls-id 15
set routing-instances titanium protocols vpls site pe1 site-identifier 1
set routing-instances titanium protocols pim-snooping
```

Router CE1

```
set interfaces ge-2/0/0 unit 0 description toPE1
set interfaces ge-2/0/0 unit 0 family inet address 10.0.0.10/30
set interfaces lo0 unit 0 family inet address 10.255.2.2./32
set routing-options router-id 10.255.2.2
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols pim rp static address 10.255.3.3
set protocols pim interface all
```

Router CE2

```
set interfaces ge-2/0/0 unit 0 description toPE1
set interfaces ge-2/0/0 unit 0 family inet address 10.0.0.6/30
set interfaces ge-2/0/1 unit 0 description toReceiver1
set interfaces ge-2/0/1 unit 0 family inet address 10.0.0.13/30
set interfaces lo0 unit 0 family inet address 10.255.2.2
set routing-options router-id 10.255.2.2
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols pim rp static address 10.255.3.3
set protocols pim interface all
```

Router PE2

```
set multicast-snooping-options traceoptions file snoop.log size 10m
set interfaces ge-2/0/0 encapsulation ethernet-vpls
set interfaces ge-2/0/0 unit 0 description toCE3
set interfaces ge-2/0/1 encapsulation ethernet-vpls
set interfaces ge-2/0/1 unit 0 description toCE4
set interfaces ge-2/0/2 unit 0 description toPE1
set interfaces ge-2/0/2 unit 0 family inet address 10.0.0.2/30
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.7.7/32
set routing-options router-id 10.255.7.7
set protocols mpls interface ge-2/0/2.0
set protocols bgp group toPE1 type internal
set protocols bgp group toPE1 local-address 10.255.7.7
set protocols bgp group toPE1 family l2vpn signaling
set protocols bgp group toPE1 neighbor 10.255.1.1
set protocols ospf area 0.0.0.0 interface ge-2/0/2.0
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ldp interface ge-2/0/2.0
set protocols ldp interface lo0.0
set routing-instances titanium instance-type vpls
set routing-instances titanium vlan-id none
set routing-instances titanium interface ge-2/0/0.0
set routing-instances titanium interface ge-2/0/1.0
set routing-instances titanium route-distinguisher 101:101
set routing-instances titanium vrf-target target:201:201
set routing-instances titanium protocols vpls vpls-id 15
set routing-instances titanium protocols vpls site pe2 site-identifier 2
set routing-instances titanium protocols pim-snooping
```

Router CE3 (RP)

```
set interfaces ge-2/0/0 unit 0 description toPE2
set interfaces ge-2/0/0 unit 0 family inet address 10.0.0.18/30
set interfaces ge-2/0/1 unit 0 description toSource
set interfaces ge-2/0/1 unit 0 family inet address 10.0.0.29/30
set interfaces lo0 unit 0 family inet address 10.255.3.3/32
set routing-options router-id 10.255.3.3
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols pim rp local address 10.255.3.3
set protocols pim interface all
```

Router CE4

```
set interfaces ge-2/0/0 unit 0 description toPE2
set interfaces ge-2/0/0 unit 0 family inet address 10.0.0.22/30
set interfaces ge-2/0/1 unit 0 description toReceiver2
set interfaces ge-2/0/1 unit 0 family inet address 10.0.0.25/30
set interfaces lo0 unit 0 family inet address 10.255.4.4/32
set routing-options router-id 10.255.4.4
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols pim rp static address 10.255.3.3
set protocols pim interface all
```

Configuring PIM Snooping for VPLS

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.

NOTE: This section includes a step-by-step configuration procedure for one or more routers in the topology. For comprehensive configurations for all routers, see "CLI Quick Configuration" on page 706.

To configure PIM snooping for VPLS:

1. Configure the router interfaces forming the links between the routers.

```
Router PE2
[edit interfaces]
user@PE2# set ge-2/0/0 encapsulation ethernet-vpls
user@PE2# set ge-2/0/0 unit 0 description toCE3
user@PE2# set ge-2/0/1 encapsulation ethernet-vpls
user@PE2# set ge-2/0/1 unit 0 description toCE4
user@PE2# set ge-2/0/2 unit 0 description toPE1
user@PE2# set ge-2/0/2 unit 0 family mpls
user@PE2# set ge-2/0/2 unit 0 family inet address 10.0.0.2/30
user@PE2# set lo0 unit 0 family inet address 10.255.7.7/32
```

NOTE: `ge-2/0/0.0` and `ge-2/0/1.0` are configured as VPLS interfaces and connect to Routers CE3 and CE4. See *Virtual Private LAN Service User Guide* for more details.

```
Router CE3
[edit interfaces]
user@CE3# set ge-2/0/0 unit 0 description toPE2
```

```
user@CE3# set ge-2/0/0 unit 0 family inet address 10.0.0.18/30
user@CE3# set ge-2/0/1 unit 0 description toSource
user@CE3# set ge-2/0/1 unit 0 family inet address 10.0.0.29/30
user@CE3# set lo0 unit 0 family inet address 10.255.3.3/32
```

NOTE: The `ge-2/0/1.0` interface on Router CE3 connects to the multicast source.

```
Router CE4
[edit interfaces]
user@CE4# set ge-2/0/0 unit 0 description toPE2
user@CE4# set ge-2/0/0 unit 0 family inet address 10.0.0.22/30
user@CE4# set ge-2/0/1 unit 0 description toReceiver2
user@CE4# set ge-2/0/1 unit 0 family inet address 10.0.0.25/30
user@CE4# set lo0 unit 0 family inet address 10.255.4.4/32
```

NOTE: The `ge-2/0/1.0` interface on Router CE4 connects to a multicast receiver.

Similarly, configure Routers PE1, CE1, and CE2.

2. Configure the router IDs of all routers.

```
Router PE2
[edit routing-options]
user@PE2# set router-id 10.255.7.7
```

Similarly, configure other routers.

3. Configure an IGP on interfaces of all routers.

```
Router PE2
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface ge-2/0/2.0
user@PE2# set interface lo0.0
```

Similarly, configure other routers.

4. Configure the LDP, MPLS, and BGP protocols on the PE routers.

   ```
   Router PE2
   [edit protocols]
   user@PE2# set ldp interface lo0.0
   user@PE2# set mpls interface ge-2/0/2.0
   user@PE2# set bgp group toPE1 type internal
   user@PE2# set bgp group toPE1 local-address 10.255.7.7
   user@PE2# set bgp group toPE1 family l2vpn signaling
   user@PE2# set bgp group toPE1 neighbor 10.255.1.1
   user@PE2# set ldp interface ge-2/0/2.0
   ```

   The BGP group is required for interfacing with the other PE router. Similarly, configure Router PE1.

5. Configure PIM on all CE routers.

   Ensure that Router CE3 is configured as the rendezvous point (RP) and that the RP address is configured on other CE routers.

   ```
   Router CE3
   [edit protocols pim]
   user@CE3# set rp local address 10.255.3.3
   user@CE3# set interface all
   ```

   ```
   Router CE4
   [edit protocols pim]
   user@CE4# set rp static address 10.255.3.3
   user@CE4# set interface all
   ```

   Similarly, configure Routers CE1 and CE2.

6. Configure multicast snooping options on the PE routers.

   ```
   Router PE2
   [edit multicast-snooping-options traceoptions]
   user@PE2# set file snoop.log size 10m
   ```

   Similarly, configure Router PE1.

7. Create a routing instance (`titanium`), and configure the VPLS on the PE routers.

```
Router PE2
[edit  routing-instances titanium]
user@PE2# set instance-type vpls
user@PE2# set vlan-id none
user@PE2# set interface ge-2/0/0.0
user@PE2# set interface ge-2/0/1.0
user@PE2# set route-distinguisher 101:101
user@PE2# set vrf-target target:201:201
user@PE2# set protocols vpls vpls-id 15
user@PE2# set protocols vpls site pe2 site-identifier 2
```

Similarly, configure Router PE1.

8. Configure PIM snooping on the PE routers.

```
Router PE2
[edit routing-instances titanium]
user@PE2# set protocols pim-snooping
```

Similarly, configure Router PE1.

Results

From configuration mode, confirm your configuration by entering the **show interfaces**, **show routing-options**, **show protocols**, **show multicast-snooping-options**, and **show routing-instances** commands.

If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE2# show interfaces
ge-2/0/2 {
    unit 0 {
        description toPE1
        family inet {
            address 10.0.0.2/30;
        }
        family mpls;
    }
}
```

```
ge-2/0/0 {
    encapsulation ethernet-vpls;
    unit 0 {
         description toCE3;
           }
}
ge-2/0/1 {
    encapsulation ethernet-vpls;
    unit 0 {
         description toCE4;
           }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.7.7/32;
        }
    }
}
```

```
user@PE2# show routing-options
router-id 10.255.7.7;
```

```
user@PE2# show protocols
mpls {
    interface ge-2/0/2.0;
}
ospf {
    area 0.0.0.0 {
        interface ge-2/0/2.0;
        interface lo0.0;
    }
}
ldp {
    interface ge-2/0/2.0;
    interface lo0.0;
}
bgp {
    group toPE1 {
        type internal;
        local-address 10.255.7.7;
        family l2vpn {
            signaling;
        }
        neighbor 10.255.1.1;
    }
```

```
user@PE2# show multicast-snooping-options
traceoptions {
    file snoop.log size 10m;
}
```

```
user@PE2# show routing-instances
titanium {
    instance-type vpls;
    vlan-id none;
    interface ge-2/0/0.0;
    interface ge-2/0/1.0;
    route-distinguisher 101:101;
    vrf-target target:201:201;
    protocols {
        vpls {
            site pe2 {
                site-identifier 2;
            }
            vpls-id 15;
        }
        pim-snooping;
  }
}
```

Similarly, confirm the configuration on all other routers. If you are done configuring the routers, enter `commit` from configuration mode.

NOTE: Use the **show protocols** command on the CE routers to verify the configuration for the PIM RP .

Verification

IN THIS SECTION

● Verifying PIM Snooping for VPLS | 715

Confirm that the configuration is working properly.

Verifying PIM Snooping for VPLS

Purpose

Verify that PIM Snooping is operational in the network.

Action

To verify that PIM snooping is working as desired, use the following commands:

- **show pim snooping interfaces**
- **show pim snooping neighbors detail**
- **show pim snooping statistics**
- **show pim snooping join**
- **show pim snooping join extensive**
- **show multicast snooping route extensive instance** ***<instance-name>*** **group** ***<group-name>***

1. From operational mode on Router PE2, run the **show pim snooping interfaces** command.

```
user@PE2> show pim snooping interfaces
Instance: titanium

Learning-Domain: default

Name                State  IP  NbrCnt
ge-2/0/0.0          Up     4        1
ge-2/0/1.0          Up     4        1
```

```
DR address: 10.0.0.22
DR flooding is ON
```

The output verifies that PIM snooping is configured on the two interfaces connecting Router PE2 to Routers CE3 and CE4.

Similarly, check the PIM snooping interfaces on Router PE1.

2. From operational mode on Router PE2, run the **show pim snooping neighbors detail** command.

```
user@PE2> show pim snooping neighbors detail
Instance: titanium
Learning-Domain: default

Interface: ge-2/0/0.0

    Address: 10.0.0.18
        Uptime: 00:17:06
        Hello Option Holdtime: 105 seconds 99 remaining
        Hello Option DR Priority: 1
        Hello Option Generation ID: 552495559
        Hello Option LAN Prune Delay: delay 500 ms override 2000 ms
                                      Tracking is supported

Interface: ge-2/0/1.0

    Address: 10.0.0.22
        Uptime: 00:15:16
        Hello Option Holdtime: 105 seconds 103 remaining
        Hello Option DR Priority: 1
        Hello Option Generation ID: 1131703485
        Hello Option LAN Prune Delay: delay 500 ms override 2000 ms
                                      Tracking is supported
```

The output verifies that Router PE2 can detect the IP addresses of its PIM snooping neighbors (10.0.0.18 on CE3 and 10.0.0.22 on CE4).

Similarly, check the PIM snooping neighbors on Router PE1.

3. From operational mode on Router PE2, run the **show pim snooping statistics** command.

```
user@PE2> show pim snooping statistics
Instance: titanium
```

```
Learning-Domain: default

Tx J/P messages                            0
RX J/P messages                          246
Rx J/P messages -- seen                    0
Rx J/P messages -- received              246
Rx Hello messages                       1036
Rx Version Unknown                         0
Rx Neighbor Unknown                        0
Rx Upstream Neighbor Unknown               0
Rx J/P Busy Drop                           0
Rx J/P Group Aggregate                     0
Rx Malformed Packet                        0

Rx No PIM Interface                        0
Rx Bad Length                              0
Rx Unknown Hello Option                    0
Rx Unknown Packet Type                     0
Rx Bad TTL                                 0
Rx Bad Destination Address                 0
Rx Bad Checksum                            0
Rx Unknown Version                         0
```

The output shows the number of hello and join/prune messages received by Router PE2. This verifies that PIM sparse mode is operational in the network.

4. Send multicast traffic from the source terminal attached to Router CE3, for the multicast group 203.0.113.1.

5. From operational mode on Router PE2, run the **show pim snooping join**, **show pim snooping join extensive**, and **show multicast snooping route extensive instance** ***<instance-name>*** **group** ***<group-name>*** commands to verify PIM snooping.

```
user@PE2> show pim snooping join
Instance: titanium
Learning-Domain: default

Group: 203.0.113.1
    Source: *
    Flags: sparse,rptree,wildcard
    Upstream neighbor: 10.0.0.18, Port: ge-2/0/0.0
```

```
Group: 203.0.113.1
    Source: 10.0.0.30
    Flags: sparse
    Upstream neighbor: 10.0.0.18, Port: ge-2/0/0.0
```

```
user@PE2> show pim snooping join extensive
Instance: titanium
Learning-Domain: default

Group: 203.0.113.1
    Source: *
    Flags: sparse,rptree,wildcard
    Upstream neighbor: 10.0.0.18, Port: ge-2/0/0.0
        Downstream port: ge-2/0/1.0
            Downstream neighbors:
                10.0.0.22 State: Join Flags: SRW Timeout: 180

Group: 203.0.113.1
    Source: 10.0.0.30
    Flags: sparse
    Upstream neighbor: 10.0.0.18, Port: ge-2/0/0.0
        Downstream port: ge-2/0/1.0
            Downstream neighbors:
                10.0.0.22 State: Join Flags: S Timeout: 180
```

The outputs show that multicast traffic sent for the group 203.0.113.1 is sent to Receiver 2 through Router CE4 and also display the upstream and downstream neighbor details.

```
user@PE2> show multicast snooping route extensive instance titanium group 203.0.113.1
Nexthop Bulking: OFF


             Family: INET

Group: 203.0.113.1/24
    Bridge-domain: titanium
    Mesh-group: __all_ces__
    Downstream interface list:
        ge-2/0/1.0 -(1072)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 1048577
    Route state: Active
    Forwarding state: Forwarding

Group: 203.0.113.1/24
    Source: 10.0.0.8
    Bridge-domain: titanium
    Mesh-group: __all_ces__
    Downstream interface list:
        ge-2/0/1.0 -(1072)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 1048577
    Route state: Active
    Forwarding state: Forwarding
```

Meaning

PIM snooping is operational in the network.

VPLS Label Blocks Operation

IN THIS SECTION

- Elements of Network Layer Reachability Information | **720**
- Requirements for NLRI Elements | **721**
- How Labels are Used in Label Blocks | **721**
- Label Block Composition | **721**
- Label Blocks in Junos OS | **722**
- VPLS Label Block Structure | **722**

A virtual private LAN service (VPLS) is a Layer 2 (L2) service that emulates a local area network (LAN) across a wide area network (WAN). VPLS labels are defined and exchanged in the Border Gateway Protocol (BGP) control plane. In the Junos OS implementation, label blocks are allocated and used in the VPLS control plane for two primary functions: autodiscovery and signaling.

- Autodiscovery—A method for automatically recognizing each provider edge (PE) router in a particular VPLS domain, using BGP update messages.

- Signaling—Each pair of PE routers in a VPLS domain sends and withdraws VPN labels to each other. The labels are used to establish and dismantle pseudowires between the routers. Signaling is also used to transmit certain characteristics of a pseudowire.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The PE router uses BGP extended communities to identify the members of its VPLS. Once the PE router discovers its members, it is able to establish and tear down pseudowires between members by exchanging and withdrawing labels and transmitting certain characteristics of the pseudowires.

The PE router sends common update messages to all remote PE routers, using a distinct BGP update message, thereby reducing the control plane load. This is achieved by using VPLS label blocks.

Elements of Network Layer Reachability Information

VPLS BGP network layer reachability information (NLRI) is used to exchange VPLS membership and parameters. The elements of a VPLS BGP NLRI are defined in Table 16 on page 720.

Table 16: NLRI Elements

Element	Acronym	Description	Default Size (Octets)
Length		Total length of the NLRI size represented in bytes.	2
Route Distinguisher	RD	Unique identifier for each routing instance configured on a PE.	8
VPLS Edge ID	VE ID	Unique number to identify the edge site.	2
VE Block Offset	VBO	Value used to identify a label block from which a label value is selected to set up pseudowires for a remote site.	2
VE Block Size	VBS	Indicates the number of pseudowires that peers can have in a single block.	2

Table 16: NLRI Elements *(Continued)*

Element	Acronym	Description	Default Size (Octets)
Label Base	LB	Starting value of the label in the advertised label block.	3

Requirements for NLRI Elements

Junos OS requires a unique route distinguisher (RD) for each routing instance configured on a PE router. A PE router might use the same RD across a VPLS (or VPN) domain or it might use different RDs. Using different RDs helps identify the originator of the VPLS NLRI.

The VPLS edge (VE) ID can be a unique VE ID, site ID, or customer edge (CE) ID. The VE ID is used by a VPLS PE router to index into label blocks used to derive the transmit and receive VPN labels needed for transport of VPLS traffic. The VE ID identifies a particular site, so it needs to be unique within the VPLS domain, except for some scenarios such as multihoming.

All PE routers have full mesh connectivity with each other to exchange labels and set up pseudowires. The VE block size (VBS) is a configurable value that represents the number of label blocks required to cover all the pseudowires for the remote peer.

A single label block contains 8 labels (1 octet) by default. The default VBS in Junos OS is 2 blocks (2 octets) for a total of 16 labels.

How Labels are Used in Label Blocks

Each PE router creates a mapping of the labels in the label block to the sites in a VPLS domain. A PE router advertising a label block with a block offset indicates which sites can use the labels to reach it. When a PE router is ready to advertise its membership to a VPLS domain, it allocates a label block and advertises the VPLS NLRI. In this way, other PE routers in the same VPLS domain can learn of the existence of the VPLS and set up pseudowires to it if needed. The VPLS NLRI advertised for this purpose is referred to as the *default VPLS NLRI*. The label block in the default VPLS NLRI is referred to as the *default label block*.

Label Block Composition

A label block (set of labels) is used to reach a given site ID. A single label block contains 8 labels (1 octet) by default. The VBS is 2 octets by default in Junos OS.

The label block advertised is defined as a label base (LB) and a VE block size (VBS). It is a contiguous set of labels (LB, LB+1,...LB+VBS-1). For example, when Router PE-A sends a VPLS update, it sends the same label block information to all other PE routers. Each PE router that receives the LB advertisement infers the label intended for Router PE-A by adding its own site ID to the label base.

In this manner, each receiving PE gets a unique label for PE-A for that VPLS. This simple method is enhanced by using a VE block offset (VBO).

A label block is defined as: <Label Base (LB), VE block offset (VBO), VE block size (VBS)> is the set {LB +VBO, LB+VBO+1,...,LB+VBO+VBS-1}.

Label Blocks in Junos OS

Instead of a single large label block to cover all VE IDs in a VPLS, the Junos OS implementation contains several label blocks, each with a different label base. This makes label block management easier, and also allows Router PE-A to seemlessly integrate a PE router joining a VPLS with a site ID not covered by the set of label blocks that Router PE-A has already advertised.

VPLS Label Block Structure

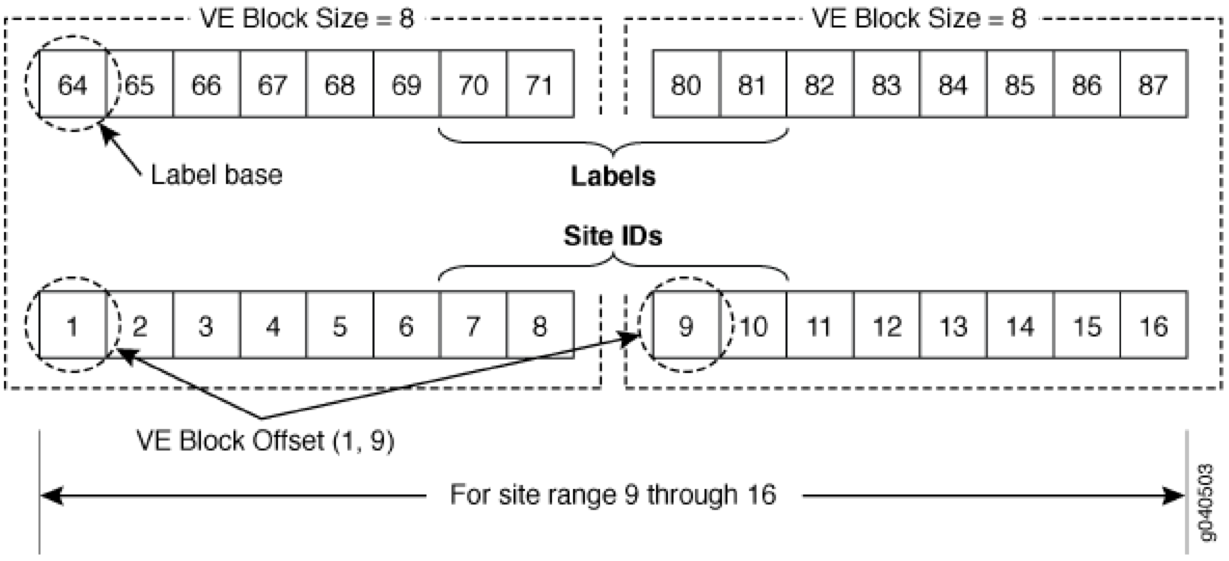
This section illustrates how a label block is uniquely identified.

A VPLS BGP NLRI with site ID V, VE block offset VBO, VE block size VBS, and label base LB communicates the following to its peers:

- Label block for V: Labels from LB to (LB + VBS -1).
- Remote VE set for V: from VBO to (VBO + VBS -1).

The label block advertised is a set of labels used to reach a given site ID. If there are several label blocks, the remote VE set helps to identify which label block to use. The example in Figure 54 on page 723 illustrates label blocks. There are two blocks and each block has eight labels. In this example, the label values are 64 to 71 and 80 to 87.

Figure 54: VPLS Label Block Structure



To create a one-to-one mapping of these 16 labels to 16 sites, assume the site IDs are the numbers 1 to 16, as shown in the illustration. The site block indicates which site ID can use which label in the label block. So, in the first block, site ID 1 uses 64, site ID 2 uses 65, and so forth. Finally, site ID 8 uses 71. The 9th site ID will use the second block instead of the first block.

The labels are calculated by comparing the values of VBO <= Local site ID < (VBO +VBS). Consequently, site ID 9 uses 80, site ID 10 uses 81, and so on.

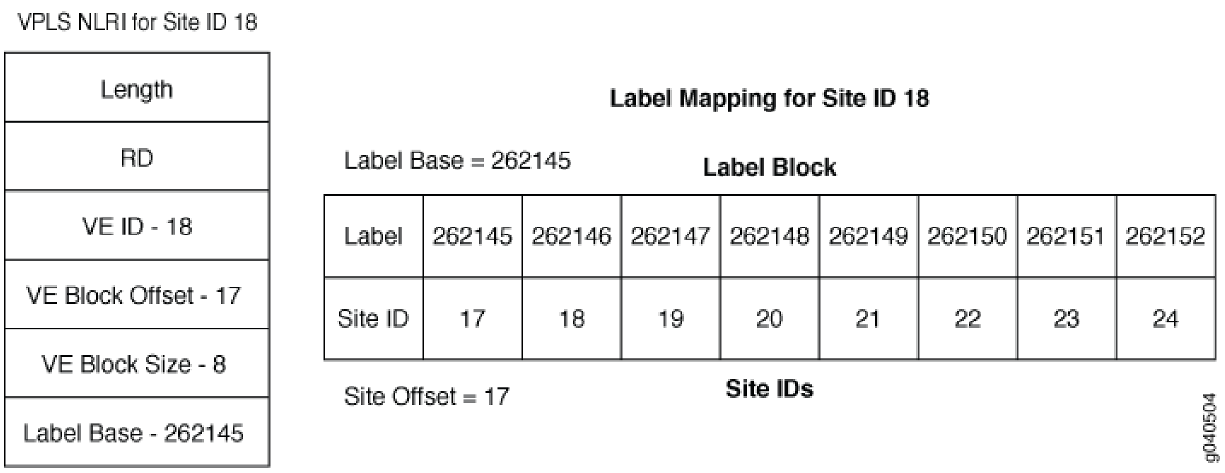
To further illustrate the one-to-one mapping of labels to sites, assume a label block with site offset of 1 and a label base of 10. The combination of label base and block offset contained in the VPLS NLRI provides the mapping of labels to site IDs. The block offset is the starting site ID that can use the label block as advertised in the VPLS NLRI.

To advertise the default VPLS NLRI, a PE router picks a starting block offset that fits its own site ID and is such that the end block offset is a multiple of a single label block. In Junos OS a single label block is eight labels by default.

The end block offset is the last site ID that maps to the last label in the label block. The end offset for the first block is 8 which maps to label 17 and the second block is 16. For example, a site with ID 3 picks a block offset of 1 and advertises a label block of size 8 to cover sites with IDs 1 to 8. A site with ID 10 picks a block offset of 9 to cover sites with IDs 9 to 16.

The VPLS NLRI shown in Figure 55 on page 724 is for site ID 18. The label base contains value 262145. The block offset contains value 17. The illustration shows which site IDs correspond to which labels.

Figure 55: Label Mapping Example



If a PE router configured with site ID 17 is in the same VPLS domain as a PE router configured with site ID 18, it receives the VPLS NLRI as shown in Figure 2. So it uses label 262145 to send traffic to site 18. Similarly, a PE router configured with site ID 19 uses label 262147 to send traffic to a PE router configured with site ID 18. However, only PE routers configured with site IDs 17 to 24 can use the label block shown to set up pseudowires.

RELATED DOCUMENTATION

Example: Building a VPLS From Router 1 to Router 3 to Validate Label Blocks | **725**

Configuring the Label Block Size for VPLS

VPLS MPLS packets have a two-label stack. The outer label is used for normal MPLS forwarding in the service provider's network. If BGP is used to establish VPLS, the inner label is allocated by a PE router as part of a label block. One inner label is needed for each remote VPLS site. Four sizes are supported. We recommend using the default size of 8, unless the network design requires a different size for optimal label usage, to allow the router to support a larger number of VPLS instances.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

If you allocate a large number of small label blocks to increase efficiency, you also increase the number of routes in the VPLS domain. This has an impact on the control plane overhead.

Changing the configured label block size causes all existing pseudowires to be deleted. For example, if you configure the label block size to be 4 and then change the size to 8, all existing label blocks of size 4 are deleted, which means that all existing pseudowires are deleted. The new label block of size 8 is created, and new pseudowires are established.

Four label block sizes are supported: 2, 4, 8, and 16. Consider the following scenarios:

- 2—Allocate the label blocks in increments of 2. For a VPLS domain that has only two sites with no future expansion plans.
- 4—Allocate the label blocks in increments of 4.
- 8 (default)—Allocate the label blocks in increments of 8.
- 16—Allocate the label blocks in increments of 16. A label block size of 16 enables you to minimize the number of routes in the VPLS domain. Use this setting only if the number of routes is the most important concern.

Use Feature Explorer to determine if a feature is supported on your platform and release.

Configure the label block size:

```
[edit routing-instances instance-name protocols vpls]
user@router# set label-block-size 2
```

RELATED DOCUMENTATION

Configuring VPLS Routing Instances | 675

Example: Building a VPLS From Router 1 to Router 3 to Validate Label Blocks

IN THIS SECTION

- Requirements | 726
- Overview and Topology | 726
- Configuration | 727

This example illustrates how VPLS label blocks are allocated for a specific configuration. It is organized in the following sections:

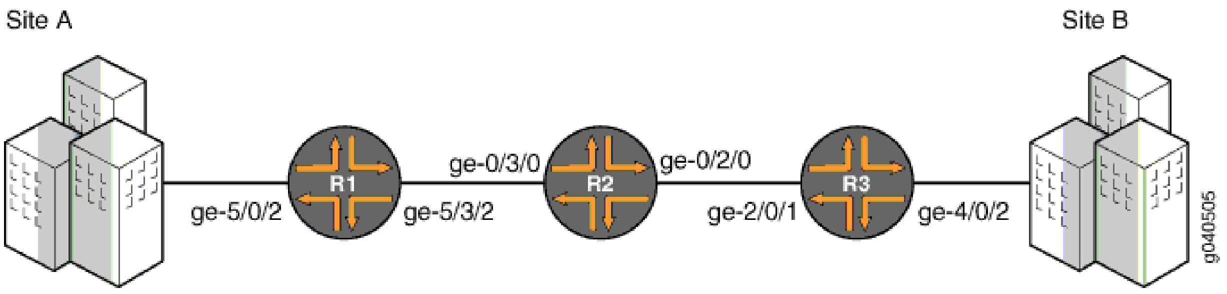
Requirements

This configuration example requires three Juniper Networks routers.

Overview and Topology

In the network shown in Figure 56 on page 726 Router 1 is establishing a pseudowire to Router 3

Figure 56: Router 1 to Router 3 Topology



Each PE filters the VPLS NLRI contained in the BGP update messages based on route target communities. Those VPLS NLRI instances that match the route target (in this case 8717:2000:2:1) are imported for further processing. The NLRI for Router 1 and Router 3 is shown in Table 17 on page 726.

Table 17: NLRI Exchange Between for Router 1 and Router 3

Router 1 NLRI Advertisement to Router 3	Router 3 NLRI Advertisement to Router 1
RD - 8717:1000	RD - 8717:1000
VE ID - 1	VE ID - 2
VE Block Offset - 1	VE Block Offset - 1
VE Block Size - 8	VE Block Size - 8
Label Base - 262161	Label Base - 262153

To set up a pseudowire to Router 3, Router 1 must select a label to use to send traffic to Router 3 and also select a label that it expects Router 3 to use to send traffic to itself. The site ID contained in the VPLS NLRI from Router 3 is 2.

Router 1 learns of the existence of site ID 2 in the same VPLS domain. Using the equation VBO <= Local Site ID < (VBO +VBS), Router 1 checks if the route advertised by site ID 2 fits in the label block and block offset that it previously advertised to Router 3. In this example it does fit, so the site ID 2 is mapped by the VPLS NLRI advertised by Router 1, and Router 1 is ready to set up a pseudowire to Router 3.

To select the label to reach Router 3, Router 1 looks at the label block advertised by Router 3 and performs a calculation. The calculation a PE router uses to check if its site ID is mapped in the label block from the remote peer is VBO <= Local Site ID < (VBO +VBS). So, Router 1 selects label (262153 + (1 -1)) = 262153 to send traffic to Router 3. Using the same equation, Router 1 looks at its own label block that it advertised and selects label (262161 + (2 - 1)) = 262162 to receive traffic from Router 3. Router 1 programs its forwarding state such that any traffic destined to Router 3 carries the pseudowire label 262153 and any traffic coming from Router 3 is expected to have the pseudowire label 262162. This completes the operations on the VPLS NLRI received from Router 3. Router 1 now has a pseudowire set up to Router 3.

Router 3 operation is very similar to the Router 1 operation. Since the Router 3 site ID of 2 fits in the label block and block offset advertised by Router 1, Router 3 selects label (262161 + (2 - 1)) = 262162 to send traffic to Router 1. Router 3 looks at its own label block that it advertised and selects label (262153 + (1 - 1)) = 262153 to receive traffic from Router 1. This completes the creation of a pseudowire to Router 1.

By default, for VPLS operation Junos OS uses a virtual tunnel (VT) loopback interface to represent a pseudowire. This example uses a label-switched interface (LSI) instead of a VT interface because there is no change in the VPLS control plane operation. Thus, for an MX platform, if there is a tunnel physical interface card (PIC) configured, it is mandatory to include the `no-tunnel-services` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level.

Configuration

IN THIS SECTION

- Configuring Router 1 | **728**
- Configuring Router 3 | **728**
- Verifying the VPLS Label Allocations | **729**

The following sections present the steps to configure and verify the example in Figure 56 on page 726.

Configuring Router 1

Step-by-Step Procedure

1. Configure Router 1. Create the `edut` routing instance. Specify the `vpls` instance type. Configure the route distinguisher and specify the value `8717:1000`. Configure the route target and specify the value `8717:100` Configure the VPLS protocol. Specify `10` as the site range. Specify `1` as the site ID. Include the `no-tunnel-services` statement.

```
[edit routing-instances]
edut {
    instance-type vpls;
    interface ge-5/0/2.0;
    route-distinguisher 8717:1000;
    vrf-target target:8717:100;
    protocols {
        vpls {
            site-range 10;
            no-tunnel-services;
            site router-1 {
                site-identifier 1;
            }
        }
    }
}
```

Configuring Router 3

Step-by-Step Procedure

1. Configure Router 3. Create the `edut` routing instance. Specify the `vpls` instance type. Configure the route distinguisher and specify the value `8717:2000`. Configure the route target and specify the value `8717:200` Configure the VPLS protocol. Specify `10` as the site range. Specify `2` as the site ID. Include the `no-tunnel-services` statement.

```
[edit routing-instances]
edut {
    instance-type vpls;
    interface ge-4/0/2.0;
    route-distinguisher 8717:2000;
    vrf-target target:8717:100;
    protocols {
        vpls {
            site-range 10;
            no-tunnel-services;
            site router-3 {
                site-identifier 2;
            }
        }
    }
}
```

Verifying the VPLS Label Allocations

Step-by-Step Procedure

1. As shown in the figure and the configuration, Site A is attached to Router 1. Site A is assigned a site ID of 1. Before Router 1 can announce its membership to VPLS `edut` using a BGP update message, Router 1 needs to allocate a default label block. In this example, the label base of the label block allocated by Router 1 is 262161. Since Router 1's site ID is 1, Router 1 associates the assigned label block with block offset of 1. The following messages aresent from Router 1 to Router 3 and displayed using the `monitor traffic interface` *`interface-name`* command:

```
user@Router1> monitor traffic interface ge-5/3/2
Jun 14 12:26:31.280818 BGP SEND 10.10.10.1+179 -> 10.10.10.3+53950
Jun 14 12:26:31.280824 BGP SEND message type 2 (Update) length 88
Jun 14 12:26:31.280828 BGP SEND flags 0x40 code Origin(1): IGP
Jun 14 12:26:31.280833 BGP SEND flags 0x40 code ASPath(2) length 0: <null>
Jun 14 12:26:31.280837 BGP SEND flags 0x40 code LocalPref(5): 100
Jun 14 12:26:31.280844 BGP SEND flags 0xc0 code Extended Communities(16): 2:8717:100
800a:19:0:0
Jun 14 12:26:31.280848 BGP SEND flags 0x90 code MP_reach(14): AFI/SAFI 25/65
Jun 14 12:26:31.280853 BGP SEND         nhop 10.10.10.1 len 4
Jun 14 12:26:31.280862 BGP SEND         8717:1000:1:1 (label base : 262161 range : 8, ce id:
1, offset: 1)
Jun 14 12:26:31.405067 BGP RECV 10.10.10.3+53950 -> 10.10.10.1+179
Jun 14 12:26:31.405074 BGP RECV message type 2 (Update) length 88
Jun 14 12:26:31.405080 BGP RECV flags 0x40 code Origin(1): IGP
Jun 14 12:26:31.405085 BGP RECV flags 0x40 code ASPath(2) length 0: <null>
Jun 14 12:26:31.405089 BGP RECV flags 0x40 code LocalPref(5): 100
Jun 14 12:26:31.405096 BGP RECV flags 0xc0 code Extended Communities(16): 2:8717:100
```

```
800a:19:0:0
Jun 14 12:26:31.405101 BGP RECV flags 0x90 code MP_reach(14): AFI/SAFI 25/65
Jun 14 12:26:31.405106 BGP RECV         nhop 10.10.10.3 len 4
Jun 14 12:26:31.405116 BGP RECV         8717:2000:2:1 (label base : 262153 range : 8, ce id:
2, offset: 1)
```

2. As shown in the figure and the configuration, Site B is attached to Router 3. Site B is assigned a site ID of 2. Before Router 3 can announce its membership to VPLS `edut` using a BGP update message, Router 3 assigns a default label block with the label base of `262153`. The block offset for this label block is 1 because its own site ID of 2 fits in the block being advertised. The following messages are sent from Router 3 to Router 1 and displayed using the `monitor traffic interface` *`interface-name`* command:

```
user@Router3> monitor traffic interface ge-2/0/1
Jun 14 12:26:31.282008 BGP SEND 10.10.10.3+53950 -> 10.10.10.1+179
Jun 14 12:26:31.282018 BGP SEND message type 2 (Update) length 88
Jun 14 12:26:31.282026 BGP SEND flags 0x40 code Origin(1): IGP
Jun 14 12:26:31.282034 BGP SEND flags 0x40 code ASPath(2) length 0: <null>
Jun 14 12:26:31.282041 BGP SEND flags 0x40 code LocalPref(5): 100
Jun 14 12:26:31.282052 BGP SEND flags 0xc0 code Extended Communities(16): 2:8717:100
800a:19:0:0
Jun 14 12:26:31.282078 BGP SEND flags 0x90 code MP_reach(14): AFI/SAFI 25/65
Jun 14 12:26:31.282088 BGP SEND         nhop 10.10.10.3 len 4
Jun 14 12:26:31.282102 BGP SEND         8717:2000:2:1 (label base : 262153 range : 8, ce id:
2, offset: 1)

Jun 14 12:26:31.283395 BGP RECV 10.10.10.1+179 -> 10.10.10.3+53950
Jun 14 12:26:31.283405 BGP RECV message type 2 (Update) length 88
Jun 14 12:26:31.283412 BGP RECV flags 0x40 code Origin(1): IGP
Jun 14 12:26:31.283419 BGP RECV flags 0x40 code ASPath(2) length 0: <null>
Jun 14 12:26:31.283426 BGP RECV flags 0x40 code LocalPref(5): 100
Jun 14 12:26:31.283435 BGP RECV flags 0xc0 code Extended Communities(16): 2:8717:100
800a:19:0:0
Jun 14 12:26:31.283443 BGP RECV flags 0x90 code MP_reach(14): AFI/SAFI 25/65
Jun 14 12:26:31.283471 BGP RECV         nhop 10.10.10.1 len 4
Jun 14 12:26:31.283486 BGP RECV         8717:1000:1:1 (label base : 262161 range : 8, ce id:
1, offset: 1)
```

3. Verify the connection status messages for Router 1 using the `show vpls connections` command. Notice the base label is `262161`, the incoming label from Router 3 is `262162`, and the outgoing label to Router 3 is `262153`.

```
user@Router1> show vpls connections instance edut extensive
Instance: edut
  Local site: router-1 (1)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    IRB interface present: no
    ge-5/0/2.0
    lsi.1049600         2         Intf - vpls edut local site 1 remote site 2
    Label-base        Offset     Range     Preference
    262161            1          8         100
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Jun 14 12:26:31 2009           1
      Remote PE: 10.10.10.3, Negotiated control-word: No
      Incoming label: 262162, Outgoing label: 262153
      Local interface: lsi.1049600, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls edut local site 1 remote site 2
    Connection History:
        Jun 14 12:26:31 2009  status update timer
        Jun 14 12:26:31 2009  loc intf up                  lsi.1049600
        Jun 14 12:26:31 2009  PE route changed
        Jun 14 12:26:31 2009  Out lbl Update                    262153
        Jun 14 12:26:31 2009  In lbl Update                     262162
        Jun 14 12:26:31 2009  loc intf down

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -< -- only outbound connection is up
CN -- circuit not provisioned    >- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
```

```
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy

Legend for interface status
Up -- operational
Dn -- down
```

4. Verify the connection status messages for Router 3 using the `show vpls connections` command. Notice the base label is `262153`, the incoming label from Router 1 is `262153`, and the outgoing label to Router 1 is `262162`.

```
user@Router3> show vpls connections instance edut extensive
Instance: edut
  Local site: router-3 (2)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    IRB interface present: no
    ge-4/0/2.0
    lsi.1050368         1         Intf - vpls edut local site 2 remote site 1
    Label-base        Offset     Range     Preference
    262153            1          8         100
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Jun 14 12:26:31 2009           1
      Remote PE: 10.10.10.1, Negotiated control-word: No
      Incoming label: 262153, Outgoing label: 262162
      Local interface: lsi.1050368, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls edut local site 2 remote site 1
    Connection History:
        Jun 14 12:26:31 2009  status update timer
        Jun 14 12:26:31 2009  loc intf up                  lsi.1050368
        Jun 14 12:26:31 2009  PE route changed
        Jun 14 12:26:31 2009  Out lbl Update                    262162
        Jun 14 12:26:31 2009  In lbl Update                     262153
        Jun 14 12:26:31 2009  loc intf down

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
```

```
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -< -- only outbound connection is up
CN -- circuit not provisioned    >- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy

Legend for interface status
Up -- operational
Dn -- down
```

RELATED DOCUMENTATION

VPLS Label Blocks Operation | 719

CHAPTER 34

Associating Interfaces with VPLS

IN THIS CHAPTER

- Configuring Interfaces for VPLS Routing | 734
- VPLS and Aggregated Ethernet Interfaces | 743
- Configuring VLAN Identifiers for VLANs and VPLS Routing Instances | 744
- Enabling VLAN Tagging | 749
- Configuring VPLS Without a Tunnel Services PIC | 749

Configuring Interfaces for VPLS Routing

IN THIS SECTION

- Configuring the VPLS Interface Name | 735
- Configuring VPLS Interface Encapsulation | 736
- Enabling VLAN Tagging | 738
- Configuring VLAN IDs for Logical Interfaces | 739
- Enabling VLANs for Hub and Spoke VPLS Networks | 740
- Sample Scenario of Hierarchical Virtual Private LAN Service on Logical Tunnel Interface | 740
- Configuring Aggregated Ethernet Interfaces for VPLS | 741

On each PE router and for each VPLS routing instance, specify which interfaces are intended for the VPLS traffic traveling between PE and CE routers. To specify the interface for VPLS traffic, include the `interface` statement in the routing instance configuration:

```
interface interface-name;
```

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`*`]`

You must also define each interface by including the following statements:

```
vlan-taggingvlan-tagging;
encapsulation encapsulation-type;
unit logical-unit-number {
    family vpls;
    vlan-id vlan-id-number;
}
```

You can include these statements at the following hierarchy levels:

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

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The following sections provide enough information to enable you to configure interfaces for VPLS routing.

Configuring the VPLS Interface Name

Specify both the physical and logical portions of the interface name, in the following format:

```
physical.logical
```

For example, in `ge-1/2/1.2`, `ge-1/2/1` is the physical portion of the interface name and `2` is the logical portion. If you do not specify the logical portion of the interface name, `0` is set by default.

A logical interface can be associated with only one routing instance.

If you enable a routing protocol on all instances by specifying `interfaces all` when configuring the master instance of the protocol at the `[edit protocols]` hierarchy level, and you configure a specific interface for VPLS routing at the `[edit routing-instances` *`routing-instance-name`*`]` hierarchy level, the latter interface statement takes precedence and the interface is used exclusively for VPLS.

If you explicitly configure the same interface name at both the `[edit protocols]` and `[edit routing-instances` *`routing-instance-name`*`]` hierarchy levels and then attempt to commit the configuration, the commit operation fails.

Configuring VPLS Interface Encapsulation

You need to specify an encapsulation type for each PE-router-to-CE-router interface configured for VPLS. This section describes the `encapsulation` statement configuration options available for VPLS.

To configure the encapsulation type on the physical interface, include the `encapsulation` statement:

```
encapsulation (ethernet-vpls | ether-vpls-over-atm-llc |extended-vlan-vpls | vlan-vpls);
```

NOTE: ACX Series routers do not support the *`ether-vpls-over-atm-llc`* and *`extended-vlan-vpls`* options for encapsulation.

You can include the `encapsulation` statement for physical interfaces at the following hierarchy levels:

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

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

You can configure the following physical interface encapsulations for VPLS routing instances:

- `ethernet-vpls`—Use Ethernet VPLS encapsulation on Ethernet interfaces that have VLAN 802.1Q tagging and VPLS enabled. The PE router expects to receive Ethernet frames with VLAN tags that are not service-delimiting. The Ethernet frames are not meaningful to the PE router and cannot be used by the service provider to separate customer traffic.

  On M Series routers (except the M320 router), the 4-port Fast Ethernet TX PIC and the 1-port, 2-port, and 4-port, 4-slot Gigabit Ethernet PICs can use the Ethernet VPLS encapsulation type.

- `ether-vpls-over-atm-llc`—For ATM intelligent queuing (IQ) interfaces only, use the Ethernet virtual private LAN service (VPLS) over ATM LLC encapsulation to bridge Ethernet interfaces and ATM interfaces over a VPLS routing instance (as described in RFC 2684, *Multiprotocol Encapsulation over ATM Adaptation Layer 5*). Packets from the ATM interfaces are converted to standard ENET2/802.3 encapsulated Ethernet frames with the frame check sequence (FCS) field removed.

- `extended-vlan-vpls`—Use extended virtual LAN (VLAN) VPLS encapsulation on Ethernet interfaces that have VLAN 802.1Q tagging and VPLS enabled and that must accept packets carrying TPIDs 0x8100, 0x9100, and 0x9901. On M Series routers (except the M320 router), the 4-port Fast Ethernet TX PIC and the 1-port, 2-port, and 4-port, 4-slot Gigabit Ethernet PICs can use the Ethernet VPLS encapsulation type.

NOTE: The built-in Gigabit Ethernet PIC on an M7i router does not support extended VLAN VPLS encapsulation.

- `vlan-vpls`—Use VLAN VPLS encapsulation on Ethernet interfaces with VLAN 802.1Q tagging and VPLS enabled. The PE router expects to receive Ethernet frames with VLAN tags that are service-delimiting. These VLAN tags can be used by the service provider to separate customer traffic. For example, LAN traffic from different customers can flow through the same service provider switch, which can then apply VLAN tags to distinguish one customer's traffic from the others. The traffic can then be forwarded to the PE router.

  Interfaces with VLAN VPLS encapsulation accept packets carrying standard TPID values only. On M Series routers (except the M320 router), the 4-port Fast Ethernet TX PIC and the 1-port, 2-port, and 4-port, 4-slot Gigabit Ethernet PICs can use the Ethernet VPLS encapsulation type.

To configure the encapsulation type for logical interfaces, include the `encapsulation` statement:

```
encapsulation (ether-vpls-over-atm-llc | vlan-vpls);
```

You can include the `encapsulation` statement for logical interfaces at the following hierarchy levels:

- `[edit interfaces *interface-name* unit *number*]`
- `[edit logical-systems *logical-system-name* interfaces *interface-name* unit *number*]`

You can configure the following logical interface encapsulations for VPLS routing instances:

- `ether-vpls-over-atm-llc`—Use Ethernet VPLS over Asynchronous Transfer Mode (ATM) logical link control (LLC) encapsulation to bridge Ethernet interfaces and ATM interfaces over a VPLS routing instance (as described in RFC 2684, *Multiprotocol Encapsulation over ATM Adaptation Layer 5*). Packets from the ATM interfaces are converted to standard ENET2/802.3-encapsulated Ethernet frames with the frame check sequence (FCS) field removed. This encapsulation type is supported on ATM intelligent queuing (IQ) interfaces only.

- `vlan-vpls`—Use VLAN VPLS encapsulation on Ethernet interfaces with VLAN 802.1Q tagging and VPLS enabled. The PE router expects to receive Ethernet frames with VLAN tags that are service-delimiting. These VLAN tags can be used by the service provider to separate customer traffic. For example, LAN traffic from different customers can flow through the same service provider switch, which can then apply VLAN tags to distinguish one customer's traffic from the others. The traffic can then be forwarded to the PE router.

Interfaces with VLAN VPLS encapsulation accept packets carrying standard TPID values only. On M Series routers (except the M320 router), the 4-port Fast Ethernet TX PIC and the 1-port, 2-port, and 4-port, 4-slot Gigabit Ethernet PICs can use the Ethernet VPLS encapsulation type.

NOTE: Label-switched interfaces (LSIs) do not support VLAN VPLS encapsulation. Therefore, you can only use VLAN VPLS encapsulation on a PE-router-to-CE-router interface and not a core-facing interface.

When you configure the physical interface encapsulation as `vlan-vpls`, you also need to configure the same interface encapsulation for the logical interface. You need to configure the `vlan-vpls` encapsulation on the logical interface because the `vlan-vpls` encapsulation allows you to configure a mixed mode, where some of the logical interfaces use regular Ethernet encapsulation (the default for logical interfaces) and some use `vlan-vpls`.

NOTE: Starting with Junos OS release 13.3, a commit error occurs when you configure `vlan-vpls` encapsulation on a physical interface and configure `family inet` on one of the logical units. Previously, it was possible to commit this invalid configuration.

SEE ALSO

Configuring VLAN and Extended VLAN Encapsulation

Enabling VLAN Tagging

Junos OS supports receiving and forwarding routed Ethernet frames with 802.1Q virtual local area network (VLAN) tags and running the Virtual Router Redundancy Protocol (VRRP) over 802.1Q-tagged interfaces. For VPLS to function properly, configure the router to receive and forward frames with 802.1Q VLAN tags by including the `vlan-tagging` statement at the `[edit interfaces` *`interface-name`*`]` hierarchy level:

```
[edit interfaces interface-name]
vlan-tagging;
```

Gigabit Ethernet interfaces can be partitioned. You can assign up to 4095 different logical interfaces, one for each VLAN, but you are limited to a maximum of 1024 VLANs on any single Gigabit Ethernet or 10-Gigabit Ethernet port. The VLAN ID range for the Gigabit Ethernet interface type is 512 through 4094.

Configuring VLAN IDs for Logical Interfaces

You can bind a VLAN identifier to a logical interface by including the `vlan-id` statement:

```
vlan-id number;
```

You can include this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number]`
- `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number]`

You can also configure a logical interface to forward packets and learn MAC addresses within each VPLS routing instance configured with a VLAN ID that matches a VLAN ID specified in a list using the `vlan-id-list` statement. VLAN IDs can be entered individually using a space to separate each ID, entered as an inclusive list separating the starting VLAN ID and ending VLAN ID with a hyphen, or a combination of both.

For example, to configure the VLAN IDs 20 and 45 and the range of VLAN IDs between 30 and 40, issue the following command from the CLI:

```
set interfaces ge-1/0/1 unit 1 vlan-id-list [20 30-40 45];
```

To configure a list of VLAN IDs for a logical interface, include the `vlan-id-list` statement:

```
vlan-id-list list-of-vlan-ids;
```

You can include this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number]`
- `[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

For more information about how to configure VLANs, see the Junos OS Network Interfaces Library for Routing Devices. For detailed information about how VLAN identifiers in a VPLS routing instance are processed and translated, see the *MX Series Layer 2 Configuration Guide*.

SEE ALSO

Junos OS Layer 2 Switching and Bridging Library

Enabling VLANs for Hub and Spoke VPLS Networks

For hub and spoke VPLS networks, you need to configure the `swap` option for the `output-vlan-map` statement on the hub facing interface of each spoke PE router. The `output-vlan-map` statement ensures that the vlan ID of the spoke PE router matches the VLAN ID of the hub PE router in the VPLS network. The following configuration example illustrates an interface configuration with the output-vlan-map statement included:

```
[edit interfaces xe-4/0/0]
vlan-tagging;
encapsulation flexible-ethernet-services;
unit 610 {
    encapsulation vlan-ccc;
    vlan-id 610;
    output-vlan-map swap;
}
```

Sample Scenario of Hierarchical Virtual Private LAN Service on Logical Tunnel Interface

This section describes how to configure the hierarchical virtual private LAN service (H-VPLS) on ACX5048 and ACX5096 routers. Junos OS for ACX5048 and ACX5096 routers supports configuring H-VPLS using the logical tunnel interface encapsulation.

For example, you have three provider edge devices (PE1, PE2 and PE3). PE2 device connects both PE1 and PE3 devices. The pseudowire connecting PE1 and PE2 devices are encapsulated with circuit cross-connect (CCC). In this case, PE1 device acts as a spoke and PE2 as a hub. The pseudowire connecting PE2 and PE3 devices are encapsulated with VPLS. You need to encapsulate CCC and VPLS pseudowires using the logical tunnel interface on the PE2 device.

The following steps describe how to encapsulate CCC and VPLS pseudowires using the logical tunnel interface on the PE2 device:

1. Create a logical tunnel interface on the PE2 device by using the `set chassis fpc` *`fpc-slot`* `pic` *`pic-slot`* `tunnel-services port` *`port-number`* CLI command. The *port-number* can be any port on the chassis which is not used for regular traffic forwarding. For example,

   ```
   [edit]
   user@host# set chassis fpc 0 pic 0 tunnel-services port 65
   ```

2. Encapsulate the CCC and VPLS pseudowires using the logical tunnel interface (lt-0/0/65) created on the PE2 device. Use the Ethernet CCC (ethernet-ccc) and Ethernet VPLS (ethernet-vpls)

encapsulation types at the [edit interfaces interface-name unit logical-unit-number] hierarchy level as shown in the example:

Device PE2

```
[edit]
user@host# set interfaces lt-0/0/65 unit 0 encapsulation ethernet-ccc;
user@host# set interfaces lt-0/0/65 unit 0 encapsulation peer-unit 1;
user@host# set interfaces lt-0/0/65 unit 1 encapsulation ethernet-vpls;
user@host# set interfaces lt-0/0/65 unit 1 encapsulation peer-unit 0;
```

3. Verify the configuration by entering the show command at the logical tunnel interface level. The output should display as follows:

```
[edit interfaces lt-0/0/65]
user@host# show
unit 0 {
    encapsulation ethernet-ccc;
    peer-unit 1;
}
unit 1 {
    encapsulation ethernet-vpls;
    peer-unit 0;
}
```

Based on this configuration, you can see that:

- The CCC pseudowire on PE2 device originates from lt-0/0/65.0
- The VPLS pseudowire on PE2 device originates from lt-0/0/65.1
- The CCC pseudowire on PE1 device originates from a regular interface
- The VPLS pseudowire on PE3 device originates from a regular interface

Configuring Aggregated Ethernet Interfaces for VPLS

You can configure aggregated Ethernet interfaces between CE devices and PE routers for VPLS routing instances. Traffic is load-balanced across all of the links in the aggregated interface. If one or more links in the aggregated interface fails, the traffic is switched to the remaining links.

For more information about how aggregated Ethernet interfaces function in the context of VPLS, see "VPLS and Aggregated Ethernet Interfaces" on page 743.

To configure aggregated Ethernet interfaces for VPLS, configure the interface for the VPLS routing instance as follows:

```
interfaces aex {
    vlan-tagging;
    encapsulation encapsulation-type;
    unit logical-unit-number {
        vlan-id number;
    }
}
```

You can configure the following physical link-layer encapsulation types for the VPLS aggregated Ethernet interface:

- `ethernet-vpls`
- `extended-vlan-vpls`
- `flexible-ethernet-services`
- `vlan-vpls`

NOTE: ACX Series routers do not support the **extended-vlan-vpls** and **vlan-vpls** encapsulation types.

For the **interface** configuration statement, in **ae*x***, the ***x*** represents the interface instance number to complete the link association; ***x*** can be from 0 through 127, for a total of 128 aggregated interfaces.

For more information about how to configure aggregated Ethernet interfaces, see the Ethernet Interfaces User Guide for Routing Devices.

The aggregated Ethernet interface must also be configured for the VPLS routing instance as shown in the following example:

```
[edit]
routing-instances {
    green {
        instance-type vpls;
        interface ae0.0;
        route-distinguisher 10.255.234.34:1;
        vrf-target target:11111:1;
        protocols {
            vpls {
                site-range 10;
                site green3 {
                    site-identifier 3;
                }
            }
        }
    }
}
```

Interface `ae0.0` represents the aggregated Ethernet interface in the routing instance configuration. The VPLS routing instance configuration is otherwise standard.

SEE ALSO

VPLS and Aggregated Ethernet Interfaces | 743

Ethernet Interfaces User Guide for Routing Devices

VPLS and Aggregated Ethernet Interfaces

You can configure aggregated Ethernet interfaces between CE devices and PE routers for VPLS routing instances. Traffic is load-balanced across all of the links in the aggregated interface. If one or more links in the aggregated interface fails, the traffic is switched to the remaining links.

Forwarding is based on a lookup of the DA MAC address. For the remote site, if a packet needs to be forwarded over an LSP, the packet is encapsulated and forwarded through the LSP. If the packet destination is a local site, it is forwarded over appropriate local site interface. For an aggregated Ethernet interface on the local site, packets are sent out of the load-balanced child interface. The Packet Forwarding Engine acquires the child link to transmit the data.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

When a received packet does not have a match to a MAC address in the forwarding database, the packet is forwarded over a set of interfaces determined from a lookup in the flooding database based on the incoming interface. This is denoted by a flood next hop. The flood next hop can include the aggregated Ethernet interface as the set of interfaces to flood the packet.

Each VPLS routing instance configured on a PE router has its own forwarding database entries that associate all of the MAC addresses the VPLS routing instance acquires with each corresponding port. A route is added to the kernel with a MAC address as the prefix and the next hop used to reach the destination. The route is an interface if the destination is local. For a remote destination, the route is a next hop for the remote site.

For local aggregated Ethernet interfaces on M Series and T Series routers, learning is based on the parent aggregated Ethernet *logical interface*. To age out MAC addresses for aggregated Ethernet interfaces, each Packet Forwarding Engine is queried to determine where the individual child interfaces are located. MAC addresses are aged out based on the age of the original interface.

For MX Series routers and EX Series switches, when a Dense Port Concentrator (DPC) learns a MAC address it causes the Routing Engine to age out the entry. This behavior applies to all logical interfaces. For an aggregated Ethernet logical interface, once all the member DPCs have aged out the entry, the entry is deleted from the Routing Engine.

RELATED DOCUMENTATION

Configuring Interfaces for VPLS Routing | **734**

Configuring Interfaces for VPLS Routing | **734**

Configuring VLAN Identifiers for VLANs and VPLS Routing Instances

You can configure VLAN identifiers for a VLAN or a VPLS routing instance in the following ways:

- By using either the `vlan-id` statement or the `vlan-tags` statement to configure a normalizing VLAN identifier. This topic describes how normalizing VLAN identifiers are processed and translated in a VLAN or a VPLS routing instance.
- By using the **input-vlan-map** and the `output-vlan-map` statements at the `[edit interfaces` *`interface-name`* `unit` *`logic-unit-number`*`]` or `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logic-unit-number`*`]` hierarchy level to configure VLAN mapping.

NOTE: In ACX5048 and ACX5096 routers, VLAN map operation is supported only if the `connectivity-type` is *ce* mode and not with *permanent* mode.

The **vlan-id** and `vlan-tags` statements are used to specify the normalizing VLAN identifier under the VLAN or VPLS routing instance. The normalizing VLAN identifier can translate or normalize the VLAN tags of packets received into a learn VLAN identifier.

NOTE: You cannot configure VLAN mapping using the **input-vlan-map** and `output-vlan-map` statements if you configure a normalizing VLAN identifier for a VLAN or VPLS routing instance using the **vlan-id** or `vlan-tags` statements.

To configure a VLAN identifier for a VLAN, include either the **vlan-id** or the `vlan-tags` statement at the `[edit interfaces` `*interface-name*` `unit` `*logic-unit-number*`] or `[edit logical-systems` `*logical-system-name*` `interfaces` `*interface-name*` `unit` `*logic-unit-number*`] hierarchy level, and then include that logical interface in the VLAN configuration.

For a VPLS routing instance, include either the **vlan-id** or `vlan-tags` statement at the `[edit interfaces` `*interface-name*` `unit` `*logic-unit-number*`] or `[edit logical-systems` `*logical-system-name*` `interfaces` `*interface-name*` `unit` `*logic-unit-number*`] hierarchy level, and then include that logical interface in the VPLS routing instance configuration.

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

NOTE: For a single VLAN or VPLS routing instance, you can include either the **vlan-id** or the `vlan-tags` statement, but not both. If you do not configure a **vlan-id** or **vlan-tags** for the VLAN or the VPLS routing instance, the Layer 2 packets received are forwarded to the outbound Layer 2 interface without having the VLAN tag modified unless an **output-vlan-map** is configured on the Layer 2 interface. This results in a frame being forwarded to a Layer 2 interface with a VLAN tag that is different from what is configured for the Layer 2 interface. Note that a frame received from the Layer 2 interface is still required to match the VLAN tag(s) specified in the interface configuration. The invalid configuration may cause a Layer 2 loop to occur. In ACX5048 and ACX5096 routers, if the interface VLAN is configured as **vlan-id-list**, it is mandatory to normalize the VPLS routing instance. **vlan-id** ***all*** is not supported in ACX5048 and ACX5096 routers.

The VLAN tags associated with the inbound logical interface are compared with the normalizing VLAN identifier. If the tags are different, they are rewritten as described in Table 19 on page 747. The source MAC address of a received packet is learned based on the normalizing VLAN identifier.

If the VLAN tags associated with the outbound logical interface and the normalizing VLAN identifier are different, the normalizing VLAN identifier is rewritten to match the VLAN tags of the outbound logical interface, as described in Table 20 on page 748.

The following steps outline the process for bridging a packet received over a Layer 2 logical interface when you specify a normalizing VLAN identifier using either the **vlan-id** ***number*** or `vlan-tags` statement for a VLAN or a VPLS routing instance:

1. When a packet is received on a physical port, it is accepted only if the VLAN identifier of the packet matches the VLAN identifier of one of the logical interfaces configured on that port.
2. The VLAN tags of the received packet are then compared with the normalizing VLAN identifier. If the VLAN tags of the packet are different from the normalizing VLAN identifier, the VLAN tags are rewritten as described in Table 19 on page 747.
3. If the source MAC address of the received packet is not present in the source MAC table, it is learned based on the normalizing VLAN identifier.
4. The packet is then forwarded toward one or more outbound Layer 2 logical interfaces based on the destination MAC address. A packet with a known unicast destination MAC address is forwarded only to one outbound logical interface. For each outbound Layer 2 logical interface, the normalizing VLAN identifier configured for the VLAN or VPLS routing instance is compared with the VLAN tags configured on that logical interface. If the VLAN tags associated with an outbound logical interface do not match the normalizing VLAN identifier configured for the VLAN or VPLS routing instance, the VLAN tags are rewritten as described in Table 20 on page 748.

The tables below show how VLAN tags are applied for traffic sent to and from the VLAN, depending on how the **vlan-id** and `vlan-tags` statements are configured for the VLAN and on how identifiers are configured for the logical interfaces in a VLAN or VPLS routing instance. Depending on your configuration, the following rewrite operations are performed on VLAN tags:

- **pop**—Remove a VLAN tag from the top of the VLAN tag stack.
- **pop-pop**—Remove both the outer and inner VLAN tags of the frame.
- **pop-swap**—Remove the outer VLAN tag of the frame and replace the inner VLAN tag of the frame.
- **swap**—Replace the VLAN tag of the frame.
- **push**—Add a new VLAN tag to the top of the VLAN stack.
- **push-push**—Push two VLAN tags in front of the frame.
- **swap-push**—Replace the VLAN tag of the frame and add a new VLAN tag to the top of the VLAN stack.
- **swap-swap**—Replace both the outer and inner VLAN tags of the frame.

Table 18 on page 747 shows the supported input and output VLAN map configurations.

Table 18: Supported Input and Output VLAN Map Configurations

Interface Type	Input-map		Output-map	
	Configuration	Parameters	Configuration	Parameters
Untagged	push	tpid.outer-vlan	pop	None
	push-push	tpid.outer-vlan/ inner-vlan	pop-pop	None
Single tagged	swap	tpid.outer-vlan	swap	tpid.outer-vlan
	push	tpid.outer-vlan	pop	None
	swap-push	tpid.outer-vlan/ inner-vlan	pop-swap	None
Dual tagged	swap	tpid.outer-vlan	swap	tpid.outer-vlan
	pop	None	push	tpid.outer-vlan
	swap-swap	tpid.outer-vlan/inner-vlan	swap-swap	tpid.outer-vlan

Table 19 on page 747 shows specific examples of how the VLAN tags for packets sent to the VLAN are processed and translated, depending on your configuration. “–” means that the statement is not supported for the specified logical interface VLAN identifier. “No operation” means that the VLAN tags of the received packet are not translated for the specified input logical interface.

Table 19: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a VLAN

VLAN Identifier of Logical Interface	VLAN Configurations for a VLAN		
	vlan-id none	vlan-id 200	vlan tags outer 100 inner 300
none	No operation	push 200	push 100, push 300

Table 19: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a VLAN *(Continued)*

VLAN Identifier of Logical Interface	VLAN Configurations for a VLAN		
	vlan-id none	vlan-id 200	vlan tags outer 100 inner 300
200	pop 200	No operation	swap 200 to 300, push 100
1000	pop 1000	swap 1000 to 200	swap 1000 to 300, push 100
vlan-tags outer 2000 inner 300	pop 2000, pop 300	pop 2000, swap 300 to 200	swap 2000 to 100
vlan-tags outer 100 inner 400	pop 100, pop 400	pop 100, swap 400 to 200	swap 400 to 300

Table 20 on page 748 shows specific examples of how the VLAN tags for packets sent from the VLAN are processed and translated, depending on your configuration. "–" means that the statement is not supported for the specified logical interface VLAN identifier. "No operation" means that the VLAN tags of the outbound packet are not translated for the specified output logical interface.

Table 20: Statement Usage and Output Rewrite Operations for VLAN Identifiers for a VLAN

VLAN Identifier of Logical Interface	VLAN Configurations for a VLAN		
	vlan-id none	vlan-id 200	vlan tags outer 100 inner 300
none	no operation	pop 200	pop 100, pop 300
200	push 200	No operation	pop 100, swap 300 to 200
1000	push 1000	swap 200 to 1000	pop 100, swap 300 to 1000

Table 20: Statement Usage and Output Rewrite Operations for VLAN Identifiers for a VLAN *(Continued)*

VLAN Identifier of Logical Interface	VLAN Configurations for a VLAN		
	vlan-id none	vlan-id 200	vlan tags outer 100 inner 300
vlan-tags outer 2000 inner 300	push 2000, push 300	swap 200 to 300, push 2000	swap 100 to 2000
vlan-tags outer 100 inner 400	push 100, push 400	swap 200 to 400, push 100	swap 300 to 400

Enabling VLAN Tagging

Junos OS supports receiving and forwarding routed Ethernet frames with 802.1Q virtual local area network (VLAN) tags and running the Virtual Router Redundancy Protocol (VRRP) over 802.1Q-tagged interfaces. For VPLS to function properly, configure the router to receive and forward frames with 802.1Q VLAN tags by including the `vlan-tagging` statement at the `[edit interfaces` *`interface-name`*`]` hierarchy level:

```
[edit interfaces interface-name]
vlan-tagging;
```

Gigabit Ethernet interfaces can be partitioned. You can assign up to 4095 different logical interfaces, one for each VLAN, but you are limited to a maximum of 1024 VLANs on any single Gigabit Ethernet or 10-Gigabit Ethernet port. The VLAN ID range for the Gigabit Ethernet interface type is 512 through 4094.

Configuring VPLS Without a Tunnel Services PIC

VPLS normally uses a dynamic virtual tunnel logical interface on a Tunnel Services PIC to model traffic from a remote site (a site on a remote PE router that is in a VPLS domain). All traffic coming from a remote site is treated as coming in over the virtual port representing this remote site, for the purposes of

Ethernet flooding, forwarding, and learning. An MPLS lookup based on the inner VPN label is done on a PE router. The label is stripped and the Layer 2 Ethernet frame contained within is forwarded to a Tunnel Services PIC. The PIC loops back the packet and then a lookup based on Ethernet MAC addresses is completed. This approach requires that the router have a Tunnel Services PIC and that the PE router complete two protocol lookups.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

You can configure VPLS without a Tunnel Services PIC by configuring the `no-tunnel-services` statement. This statement creates a label-switched interface (LSI) to provide VPLS functionality. An LSI MPLS label is used as the inner label for VPLS. This label maps to a VPLS routing instance. On the PE router, the LSI label is stripped and then mapped to a logical LSI interface. The Layer 2 Ethernet frame is then forwarded using the LSI interface to the correct VPLS routing instance.

By default, VPLS requires a Tunnel Services PIC. To configure VPLS on a router without a Tunnel Services PIC and create an LSI, include the `no-tunnel-services` statement:

```
no-tunnel-services;
```

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

To configure a VPLS routing instance on a router without a tunnel services PIC, include the `no-tunnel-services` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level. To configure static VPLS on a router without a tunnel services PIC, include the `no-tunnel-services` statement at the `[edit protocols vpls static-vpls]` hierarchy level.

When you configure VPLS without a Tunnel Services PIC by including the `no-tunnel-services` statement, the following limitations apply:

- An Enhanced FPC is required.
- Channelized interfaces are not supported as core-facing interfaces.
- GRE-encapsulated interfaces are not supported as core-facing interfaces.

RELATED DOCUMENTATION

Configuring Static Pseudowires for VPLS | 751

CHAPTER 35

Configuring Pseudowires

IN THIS CHAPTER

- Configuring Static Pseudowires for VPLS | **751**
- VPLS Path Selection Process for PE Routers | **753**
- BGP and VPLS Path Selection for Multihomed PE Routers | **755**
- Dynamic Profiles for VPLS Pseudowires | **757**
- Use Cases for Dynamic Profiles for VPLS Pseudowires | **758**
- Example: Configuring VPLS Pseudowires with Dynamic Profiles—Basic Solutions | **759**
- Example: Configuring VPLS Pseudowires with Dynamic Profiles—Complex Solutions | **764**
- Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | **770**
- Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | **772**
- Example: Configuring H-VPLS BGP-Based and LDP-Based VPLS Interoperation | **774**
- Example: Configuring BGP-Based H-VPLS Using Different Mesh Groups for Each Spoke Router | **805**
- Example: Configuring LDP-Based H-VPLS Using a Single Mesh Group to Terminate the Layer 2 Circuits | **833**
- Example: Configuring H-VPLS With VLANs | **842**
- Example: Configuring H-VPLS Without VLANs | **860**
- Configure Hot-Standby Pseudowire Redundancy in H-VPLS | **876**
- Sample Scenario of H-VPLS on ACX Series Routers for IPTV Services | **881**

Configuring Static Pseudowires for VPLS

You can configure a VPLS domain using static pseudowires. A VPLS domain consists of a set of PE routers that act as a single virtual Ethernet bridge for the customer sites connected to these routers. By configuring static pseudowires for the VPLS domain, you do not need to configure the LDP or BGP protocols that would normally be used for signaling. However, if you configure static pseudowires, any changes to the VPLS network topology have to be managed manually.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Static pseudowires require that you configure a set of in and out labels for each pseudowire configured for the VPLS domain. You still need to configure a VPLS identifier and neighbor identifiers for a static VPLS domain. You can configure both static and dynamic neighbors within the same VPLS routing instance.

To configure a static pseudowire for a VPLS neighbor, include the `static` statement:

```
static (Protocols VPLS) {
    incoming-label label;
    outgoing-label label;
}
```

You must configure an incoming and outgoing label for the static pseudowire using the `incoming-label` and `outgoing-label` statements. These statements identify the static pseudowire's incoming traffic and destination.

To configure a static pseudowire for a VPLS neighbor, include the `static` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`*`]` hierarchy level.

You can also configure the `static` statement for a backup neighbor (if you configure the neighbor as static the backup must also be static) by including it at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`* `backup-neighbor` *`address`*`]` hierarchy level.

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

To enable static VPLS on a router, you need to either configure a virtual tunnel interface (requires the router to have a tunnel services PIC) or you can configure a label switching interface (LSI). To configure an LSI, include the `no-tunnel-services` statement at the `[edit protocols vpls static-vpls]` hierarchy level. For more information, see "Configuring VPLS Without a Tunnel Services PIC" on page 749.

NOTE: Static pseudowires for VPLS using an LSI is supported on MX series routers and EX Series switches only. For M series and T series routers, a tunnel services PIC is required.

If you issue a `show vpls connections` command, static neighbors are displayed with `"SN"` next to their addresses in the command output.

RELATED DOCUMENTATION

Configuring VPLS Without a Tunnel Services PIC | 749

VPLS Path Selection Process for PE Routers

The VPLS path selection process is used to select the best path between a remote PE router and a local PE router in a VPLS network. This path selection process is applied to routes received from both single-homed and multi-homed PE routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

When the VPLS path selection process is complete, a PE router is made the designated VPLS edge (VE) device. The designated VE device effectively acts as the endpoint for the VPLS pseudowire that is signaled from the remote PE router. Once a PE router is made the designated VE device, a pseudowire can be signaled between the remote PE router and the local PE router and then VPLS packets can begin to flow between the PE routers.

Routes from multihomed PE routers connected to the same customer site share the same site ID, but can have different route distinguishers and block offsets. You can alter the configurations of the route distinguishers and block offsets to make a router more likely or less likely to be selected as the designated VE device.

On each PE router in the VPLS network, the best path to the CE device is determined by completing the following VPLS path selection process on each route advertisement received:

1. If the advertisement has the down bit set to 0, the advertisement is discarded.

2. Select the path with a higher preference. The preference attribute is obtained from the site-preference configured using the `site-preference` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]` hierarchy level. When the site is down, the preference attribute is obtained from the local preference.

3. If the preference values are the same, select the path with the lower router ID.

4. If the router IDs are the same, the routes are from the same PE router and the advertisement is considered to be an update. The router ID corresponds to the value of the originator ID for the BGP attribute (if present). Otherwise, the IP address for the remote BGP peer is used.

5. If the block offset values are the same, the advertisement is considered to be an update.

Once the VPLS path selection process has been completed and the designated VE device has been selected, a pseudowire is signaled between the remote PE router and the local PE router.

NOTE: The VPLS path selection process works the same whether or not the route has been received from another PE router, a route reflector, or an autonomous system border router (*ASBR*).

When the remote PE router establishes or refreshes a pseudowire to the local PE router, it verifies that the prefix is in the range required for the site ID based on the block offset and label range advertised by the designated VE device. If the prefix is out of range, the pseudowire status is set to out of range.

The following cases outline the potential decisions that could be made when a PE router completes the VPLS path selection process for a Layer 2 advertisement in the VPLS network:

- The PE router originated one of the advertisements and selected its own advertisement as the best path.

  This PE router has been selected as the designated VE device. Selection as the designated VE device triggers the creation of pseudowires to and from the other PE routers in the VPLS network. If the remote customer site is multihomed, the designated VE device triggers the creation of pseudowires to and from only the designated VE device for the remote site.

- The PE router originated one of the advertisements but did not select its own advertisement as the best path.

  This PE router is a redundant PE router for a multihomed site, but it was not selected as the designated VE device. However, if this PE router has just transitioned from being the designated VE device (meaning it was receiving traffic from the remote PE routers addressed to the multihomed customer site), the PE router tears down all the pseudowires that it had to and from the other PE routers in the VPLS network.

- The PE router received the route advertisements and selected a best path. It did not originate any of these advertisements because it was not connected to the customer site.

  If the best path to the customer site (the designated VE device) has not changed, nothing happens. If the best path has changed, this PE router brings up pseudowires to and from the newly designated VE device and tears down the pseudowires to and from the previously designated VE device.

  If this PE router does not select a best path after running the VPLS path selection process, then the local PE router does not consider the remote site to exist.

When a VE device receives an advertisement for a Layer 2 NLRI that matches its own site ID but the site is not multihomed, the pseudowire between the VE device and the transmitting PE router transitions to a site collision state and is not considered to be up.

RELATED DOCUMENTATION

BGP Route Reflectors for VPLS | **628**

BGP and VPLS Path Selection for Multihomed PE Routers

The BGP and VPLS path selection procedures are used to select the best path between the remote PE router and one of the multihomed PE routers. As part of these path selection procedures, one of the multihomed PE routers is made the designated VE device. The designated VE device effectively acts as the endpoint for the VPLS pseudowire from the remote PE router. Once a multihomed PE router is made the designated VE device, a pseudowire can be created between the remote PE router and the multihomed PE router.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Routes from multihomed PE routers connected to the same customer site share the same site ID, but can have different route distinguishers and block offsets. On each PE router in the VPLS network, the best path to the multihomed PE router is determined by completing the following VE device-selection procedures on each route advertisement received from a multihomed PE router:

1. BGP designated VE device-selection procedure—Runs before the VPLS designated VE device-selection procedure. However, the BGP designated VE device-selection procedure is used only when the route distinguishers for the multihomed PE routers are identical. If the route distinguishers are unique, only the VPLS designated VE device-selection procedure is run.

2. VPLS designated VE device-selection procedure—Runs after the BGP designated VE device-selection procedure. However, if the route distinguishers for each multihomed PE router are unique, the advertisements are not considered relevant to the BGP designated VE device-selection procedure. As a consequence, only the VPLS designated VE device-selection procedure is used.

The BGP designated VE device-selection procedure is as follows:

1. If the advertisement has the down bit set to 0, the advertisement is discarded.

2. Select the path with a higher preference. The preference attribute is obtained from the site-preference configured using the `site-preference` statement at the [edit routing-instances *routing-instance-name* protocols vpls site *site-name*] hierarchy level. If the site-preference is 0, the preference attribute is obtained from the local-preference.

3. If the preference values are the same, select the path with the lower router-id.

4. If the router-ids are the same, the routes are from the same PE router and the advertisement is considered to be an update.

Once the BGP designated VE device-selection procedure is complete, the VPLS designated VE device-selection procedure begins. This procedure is carried out regardless of the outcome of the BGP designated VE device-selection procedure:

1. If the advertisement has the down bit set to 0, the advertisement is discarded.

2. Select the path with a higher preference. The preference attribute is obtained from the site-preference configured using the `site-preference` statement at the [edit routing-instances *routing-instance-name* protocols vpls site *site-name*] hierarchy level. If the site-preference is 0, the preference attribute is obtained from the local-preference.

3. If the preference values are the same, select the path with the lower router-id.

4. If the router-ids are the same, select the path with a lower route distinguisher.

5. If the route distinguishers are the same, select the path with the lower block offset value.

6. If the block offset values are the same, the advertisement is considered to be an update.

Once the BGP and VPLS path selection procedures have been completed and the designated VE devices have been selected, a pseudowire can be created between the remote PE router and the multihomed PE router.

When the remote PE router establishes or refreshes a pseudowire to the local PE router, it verifies that the prefix is in the range required for the site ID based on the block offset and label range advertised by the designated VE device. If the prefix is out of range, the pseudowire status is set to out of range.

The following cases outline the potential decisions that could be made when a PE router completes the BGP and VPLS path selection procedures for a Layer 2 advertisement in the VPLS network:

- The PE router originated one of the multihomed advertisements and selected its own advertisement as the best path.

  This PE router has been selected as the designated VE device. Selection as the designated VE device triggers the creation of pseudowires to and from the other PE routers in the VPLS network. When the remote customer site is also multihomed, the designated VE device triggers the creation of pseudowires to and from only the designated VE device for the remote site.

- The PE router originated one of the multihomed advertisements but did not select its own advertisement as the best path.

  This PE router is one of the redundant PE routers for the multihomed site; it was not selected as the designated VE device. However, if this PE router has just transitioned from being the designated VE device (meaning it was receiving traffic from the remote PE routers addressed to the multihomed customer site), the PE router tears down all the pseudowires that it had to and from the other PE routers in the VPLS network.

- The PE router receives the multihomed advertisements and selects a best path; it does not originate any of these advertisements because it is not connected to the multihomed customer site.

  If the preferred path to the customer site (the designated VE device) has not changed, nothing happens. If the preferred path has changed, this PE router brings up pseudowires to and from the newly designated VE device and tears down the pseudowires to and from the previously designated VE device.

  If this PE router does not select a best path after running the BGP and VPLS path selection process, the local PE router does not consider the remote site to exist.

When a VE device receives an advertisement for a Layer 2 NLRI which matches its own site ID but the site is not multihomed, the pseudowire between it and the transmitting PE router transitions to a site collision state and is not considered to be up.

Dynamic Profiles for VPLS Pseudowires

A router often has two types of interfaces:

- Static interfaces, which are configured before the router is booted
- Dynamic interfaces, which are created after the router is booted and while it is running

A virtual private LAN service (VPLS) pseudowire interface (such as `lsi.1048576`) is dynamically created by the system. Therefore, the *logical interface* unit number for the VPLS pseudowire is not available in advance to configure characteristics such as virtual local area network (VLAN) identifiers and other parameters. As a result, certain VLAN manipulation features that are easily applied to static interfaces (such as `xe-`, `ge-`, and so on) are either not supported on dynamic interfaces or supported in a nonstandard method.

However, on MX Series routers, there is another configuration method that dynamic interfaces can use to determine their VLAN parameters when they are created by a running router: *dynamic profiles*. A dynamic profile is a conceptual container that includes parameters associated with a dynamic entity, parameters whose values are not known at the time the entity is configured.

A dynamic profile acts as a kind of template that enables you to create, update, or remove a configuration that includes client access (for example, interface or protocol) or service (for example, CoS) attributes. Using these profiles you can consolidate all of the common attributes of a client (and eventually a group of clients) and apply the attributes simultaneously. The router contains several predefined variables that enable dynamic association of interfaces and logical units to incoming subscriber requests. While configuring a dynamic profile, use the `$junos-interface-ifd-name` variable for a dynamic physical interface and the `$junos-underlying-unit-number` variable for a dynamic logical interface (unit). When a client accesses the router, the dynamic profile configuration replaces the predefined variable with the actual interface name or unit value for the interface the client is accessing.

Dynamic profiles for VPLS are supported only on MX Series routers. For more information about dynamic profiles, see Junos OS Broadband Subscriber Management and Services Library.

RELATED DOCUMENTATION

Ethernet Networking User Guide for MX Series Routers

Dynamic Profiles Overview

Use Cases for Dynamic Profiles for VPLS Pseudowires | **758**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Basic Solutions | **759**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Complex Solutions | **764**

Use Cases for Dynamic Profiles for VPLS Pseudowires

A dynamic profile is a set of characteristics, defined in a type of template, that you can use to provide dynamic subscriber access and services for broadband applications. These services are assigned dynamically to interfaces. You can use dynamic profiles to configure the VLAN parameters of the dynamic interfaces on MX Series routers.

Two use cases for configuring VPLS pseudowires with dynamic profiles are:

- **Configuring an extra VLAN tag onto pseudowire traffic** — This is a common scenario where all the traffic received from a customer edge (CE) interface needs an additional VLAN tag toward the core. In such cases, you can use dynamic profiles at ingress and egress to control the pseudowire behavior. You can apply dynamic profiles to receive the desired frames, set the additional VLAN tag for these frames, and send these tagged frames with the desired VLAN identifier.

- **Configuring a VPLS pseudowire as a trunk interface** — This is another common scenario where the requirement is to accept traffic from a particular source and to route the traffic based on specific criteria. Dynamic profiles can be used to create multiple pseudowire trunk interfaces to accept the traffic based on specific VLAN identifiers, and to route the accepted traffic to the desired destination.

RELATED DOCUMENTATION

Dynamic Profiles Overview

Dynamic Profiles for VPLS Pseudowires | **757**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Basic Solutions

IN THIS SECTION

- VPLS Pseudowire Interfaces Without Dynamic Profiles | 759
- VPLS Pseudowire Interfaces and Dynamic Profiles | 760
- CE Routers Without Dynamic Profiles | 762
- CE Routers and Dynamic Profiles | 763

The following limitations apply to dynamic profiles for VPLS on MX Series routers:

- The `native-vlan-id` statement is not supported.
- The `native-inner-vlan-id` statement is not supported.
- The `interface-mode access` statement option is not supported.
- The `vlan-id-range` statement is not supported.

In many cases, a configuration using dynamic profiles is more efficient than a static configuration, as shown by the examples in this topic.

VPLS Pseudowire Interfaces Without Dynamic Profiles

Consider the following configuration, which does not use dynamic profiles to manipulate VLAN identifiers:

```
[edit routing-instances]
green {
    instance-type vpls;
    interface ge-0/0/1.1;
    interface ge-0/0/2.1;
    interface ge-0/0/3.1;
    vlan-tags outer 200 inner 100;
    protocols vpls {
        vpls-id 10;
        neighbor 10.1.1.20;
    }
    {...more...}
}

[edit interfaces]
ge-0/0/1 {
    unit 0 {
        vlan-id 10;
    }
}
ge-0/0/2 {
    unit 0 {
        vlan-id 20;
    }
}
ge-0/0/3 {
    unit 0 {
        vlan-id 30;
    }
}
```

NOTE: This is not a complete router configuration.

With this configuration, broadcast packets inside frames arriving with VLAN identifier 10 on **ge-0/0/1** are normalized to a dual-tagged frame with an outer VLAN value of 200 and an inner VLAN value of 100. The broadcast packet and frames exiting **ge-0/0/2** or **ge-0/0/3** have the outer VLAN value stripped and the inner VLAN value swapped to 20 and 30 respectively, according to the interface configuration. However, this stripping of the outer VLAN tag and the swapping is extra work, because the frames will still egress the VPLS pseudowire in routing instance **green** with an outer VLAN tag value of 200 and an inner VLAN tag value of 100, also according to the configuration.

The same configuration can be accomplished more effectively using dynamic profiles.

VPLS Pseudowire Interfaces and Dynamic Profiles

Consider the following configuration, which uses dynamic profiles to manipulate VLAN identifiers:

```
[edit routing-instances]
green {
    instance-type vpls;
    interface ge-0/0/1.1;
    interface ge-0/0/2.1;
    interface ge-0/0/3.1;
    vlan-id 100; # Desired inner VLAN tag on the VPLS pseudowire
    protocols vpls {
        vpls-id 10;
        neighbor 10.1.1.20 {
            associate-profile green_vpls_pw_1; # The profile
        }
    }
    {...more...}
}

[edit interfaces]
ge-0/0/1 {
    unit 0 {
        vlan-id 10;
    }
}
ge-0/0/2 {
    unit 0 {
        vlan-id 20;
    }
}
ge-0/0/3 {
    unit 0 {
        vlan-id 30;
    }
}

[edit dynamic-profiles]
green_vpls_pw_1 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        vlan-tags outer 200 inner 100;
    }
}
```

NOTE: This is not a complete router configuration.

With this configuration, broadcast packets inside frames arriving with VLAN identifier 10 on **ge-0/0/1** normalized to a frame with VLAN identifier 100. The broadcast packet and frames exiting **ge-0/0/2** or **ge-0/0/3** have this VLAN value swapped to 20 and 30 respectively, according to the interface configuration. Frames egress the VPLS pseudowire in routing instance **green** with an outer VLAN tag value of 200 pushed on top of the normalized value.

CE Routers Without Dynamic Profiles

You can apply a dynamic profile to an entire VPLS configuration, not just a neighbor.

Consider the following configuration, which does not use dynamic profiles to manipulate VLAN identifiers on a customer edge (CE) router with VLAN identifier 100:

```
[edit routing-instances]
green {
    instance-type vpls;
    interface ge-0/0/1.1;
    interface ge-0/0/2.1;
    interface ge-0/0/3.1;
    vlan-tags outer 200 inner 100;
    protocols vpls {
        vpls-id 10;
        neighbor 10.1.1.20;
    }
    {...more...}
}

[edit interfaces]
ge-0/0/1 {
    unit 0 {
        vlan-id 100;
    }
}
ge-0/0/2 {
    unit 0 {
        vlan-id 100;
    }
}
ge-0/0/3 {
    unit 0 {
        vlan-id 100;
    }
}
```

NOTE: This is not a complete router configuration.

With this configuration, broadcast packets inside frames arriving on **ge-0/0/1** are normalized to a dual-tagged frame with an outer VLAN value of 200 and an inner VLAN value of 100. The same configuration can be accomplished using dynamic profiles.

CE Routers and Dynamic Profiles

Consider the following configuration, which uses dynamic profiles at the **protocols** level:

```
[edit routing-instances]
green {
    instance-type vpls;
    interface ge-0/0/1.1;
    interface ge-0/0/2.1;
    interface ge-0/0/3.1;
    vlan-id 100; # Desired inner VLAN tag on the VPLS pseudowire
    protocols vpls {
        associate-profile green_vpls_pw_2; # The profile
        vpls-id 10;
        neighbor 10.1.1.20;
    }
    {...more...}
}

[edit interfaces]
ge-0/0/1 {
    unit 0 {
        vlan-id 100;
    }
}
ge-0/0/2 {
    unit 0 {
        vlan-id 100;
    }
}
ge-0/0/3 {
    unit 0 {
        vlan-id 100;
    }
}

[edit dynamic-profiles]
green_vpls_pw_2 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        vlan-tags outer 200 inner 100;
    }
}
```

NOTE: This is not a complete router configuration.

With this configuration, broadcast packets inside frames arriving with VLAN identifier 100 on **ge-0/0/1** are normalized to a frame with VLAN identifier 100 (in this case, they are unchanged). The broadcast packet and frames exiting **ge-0/0/2** or **ge-0/0/3** are unchanged as well, according to the interface configuration. Frames egress the VPLS pseudowire in routing instance **green** with an outer VLAN tag value of 200 pushed on top of the normalized value.

RELATED DOCUMENTATION

https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-vpns/config-guide-vpns-layer-2.html

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Complex Solutions | **764**

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Complex Solutions

IN THIS SECTION

- Configuration of Routing Instance and Interfaces Without Dynamic Profiles | **765**
- Configuration of Routing Instance and Interfaces Using Dynamic Profiles | **766**
- Configuration of Tag Translation Using Dynamic Profiles | **769**

Dynamic profiles for VPLS pseudowires can be helpful in a variety of VLAN configurations. This section explores some of these situations through examples.

NOTE: These examples are not complete router configurations.

All of the examples in this section address the same basic topology. A routing instance **blue** uses a trunk bridge to connect different departments in an organization, each with their own VLANs, at two different sites. The organization uses a BGP-based VPLS with a virtual switch to accomplish this.

Configuration of Routing Instance and Interfaces Without Dynamic Profiles

The basic configuration of routing instance and interfaces without dynamic profiles follows:

```
[edit routing-instance blue]
instance-type virtual-switch;
route-distinguisher 10.1.1.10:1;
vrf-target target:1000:1;
interface ge-3/0/0; # The trunk interface
bridge-domains {
    sales {
        vlan-id 10;
        interface ge-0/0/0.1;
        ... # Other interfaces and statements for Sales
    }
    engineering {
        vlan-id 20;
        interface ge-1/0/2.0;
        ... # Other interfaces and statements for Engineering
    }
    accounting {
        vlan-id 30;
        interface ge-2/0/3.0;
        ... # Other interfaces and statements for Accounting
    }
    others {
        vlan-id—list [ 40 50 ]; # Other departments
    }
}
    protocols vpls {
        site-range 10;
        site sample-site-1 {
            site-identifier 1;
        }
    }
... # Other statements for instance Blue

[edit interfaces]
ge-0/0/1 {
    unit 0 {
        vlan-id 100;
    }
}
ge-3/0/0 {
    unit 0 {
        family bridge {
            interface-mode trunk; # This is the trunk
            vlan-id-list [ 10 20 30 40 50 ];
        }
    }
}
... # More interface statements
```

This configuration switches the departmental VLAN traffic (sales, engineering, etc.) bridge domains over the VPLS pseudowire trunk connecting to the other site.

Configuration of Routing Instance and Interfaces Using Dynamic Profiles

Here is how dynamic profiles can be applied to this basic configuration.

First, consider the requirement to push an outer VLAN tag value of 200 onto the VPLS pseudowire frames on egress. Dynamic profiles easily satisfy this requirement.

```

[edit routing-instance green]
instance-type virtual-switch;
... # Other routing instance statements
protocols vpls {
    site-range 10;
    site sample-site-1 {
        site-identifier 1;
    }
    associate-profile green_vpls_pw_1; # Apply profile here
}
... # Other routing instance statements

[edit dynamic-profiles]
```

```
green_vpls_pw_1 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        vlan-id 200; # This is the outer tag
        family bridge {
            interface-mode trunk;
            inner-vlan-id-list [ 10 20 30 40 50 ];
        }
    }
}
```

NOTE: This is not a complete router configuration.

With the dynamic profile, a packet in a frame arriving on an interface is classified as belonging to one of the bridge domains (VLANs 10–50). At the egress of the trunk VPLS pseudowire, the outer VLAN tag 200 is pushed onto the frame. At the ingress of the pseudowire at the remote location, the outer VLAN tag 200 is removed and the frame is delivered to the appropriate bridge domain.

But what if the packets associated with the Accounting VLAN are not to be forwarding to the remote site? Dynamic profiles are useful here as well.

This configuration keeps the Accounting frames from reaching the remote site.

```
[edit routing-instances green]
instance-type virtual-switch;
... # Other routing instance statements
protocols vpls {
    site-range 10;
    site sample-site-1 {
        site-identifier 1;
    }
    associate-profile green_vpls_pw_2; # Apply profile here
}
... # Other routing instance statements

[edit dynamic-profiles]
green_vpls_pw_2 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        family bridge {
            interface-mode trunk;
            inner-vlan-id-list [ 10 20 40 50 ]; # Removed Accounting VLAN 30
        }
    }
}
```

NOTE: This is not a complete router configuration.

In this case, frames arriving on the interfaces are classified according to their bridge domains and switched, if necessary, to the VPLS pseudowire trunk, except for Engineering frames. Engineering frames (VLAN 30) are only switched within the interfaces listed within bridge domain **accounting** and any statically configured trunk interfaces and are prevented from crossing the VPLS pseudowire due to the absence of VLAN 30 on the trunk.

We can combine the two examples and use dynamic profiles to forward the frames (other than **accounting** frames) to the remote site with an out tag of 200.

This configuration keeps the Accounting frames from reaching the remote site and pushes an outer tag of 200 on VPLS pseudowire traffic.

```
[edit routing-instances green]
instance-type virtual-switch;
... # Other routing instance statements
protocols vpls {
    site-range 10;
    site sample-site-1 {
        site-identifier 1;
    }
    associate-profile green_vpls_pw_3; # Apply profile here
}
... # Other routing instance statements

[edit dynamic-profiles]
green_vpls_pw_3 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        vlan-id 200; # This is the outer tag
        family bridge {
            interface-mode trunk;
            inner-vlan-id-list [ 10 20 40 50 ]; # Removed Accounting VLAN 30
        }
    }
}
```

NOTE: This is not a complete router configuration.

In this case, frames arriving on the interfaces are classified according to their bridge domains and switched, if necessary, to the VPLS pseudowire trunk with an outer VLAN tag of 200, except for Engineering frames. Engineering frames (VLAN 30) are only switched within the interfaces listed within bridge domain **accounting** and any statically configured trunk interfaces and are prevented from crossing the VPLS pseudowire due to the absence of VLAN 30 on the trunk.

Configuration of Tag Translation Using Dynamic Profiles

Consider a final case where the bridge domain VLANs need translation at the VPLS pseudowire trunk interface. In this case, **sales** (VLAN 10) is mapped to VLAN 110 and **engineering** (VLAN 20) is mapped to VLAN 120.

This configuration adds tag translation to the VPLS pseudowire traffic.

```
[edit routing-instances green]
instance-type virtual-switch;
... # Other routing instance statements
protocols vpls {
    site-range 10;
    site sample-site-1 {
        site-identifier 1;
    }
    associate-profile green_vpls_pw_4; # Apply profile here
}
... # Other routing instance statements

[edit dynamic-profiles]
green_vpls_pw_4 interfaces $junos-interface-ifd-name {
    unit $junos-underlying-unit-number {
        family bridge {
            interface-mode trunk;
            vlan-id-list [ 10 20 30 40 50 ]; # All VLANs
            vlan-rewrite translate 110 10; # Sales VLAN
            vlan-rewrite translate 120 20; # Engineering VLAN
        }
    }
}
```

NOTE: This is not a complete router configuration.

This translates the **sales** and **engineering** VLAN tags exiting the VPLS pseudowire accordingly. At the ingress of the VPLS pseudowire, VLANs 110 and 120 are translated back to 10 and 20, respectively.

RELATED DOCUMENTATION

https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-vpns/config-guide-vpns-layer-2.html

Example: Configuring VPLS Pseudowires with Dynamic Profiles—Basic Solutions | **759**

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic

This topic shows how to configure flow-aware transport of pseudowires (FAT) flow labels for forwarding equivalence class (FEC) 128 virtual private LAN service (VPLS) pseudowires for load-balancing MPLS traffic.

Before you begin:

1. Configure the device interfaces and enable MPLS on all the interfaces on the provider edge (PE) router.
2. Configure MPLS and an LSP from the ingress PE router to the remote egress PE router.
3. Configure an interior gateway protocol (IGP) on the PE router and provider (P) routers.
4. Configure the circuits between the PE routers and the customer edge (CE) routers.
5. Configure LDP on all the interfaces.

FAT flow labels enable load-balancing of MPLS packets across equal-cost multipath (ECMP) paths or link aggregation groups (LAGs) without the need for deep packet inspection of the payload. FAT flow labels can be used for LDP-signaled FEC 128 and FEC 129 pseudowires for VPLS and virtual private wire service (VPWS) networks.

You can configure FAT flow labels to be signaled by LDP on FEC 128 VPLS pseudowires by including the `flow-label-transmit` and `flow-label-receive` configuration statements at the `[edit routing-instances` *`instance-name`* `protocols vpls]` hierarchy level. This configuration sets the T bit and R bit advertisement to 1 (the default being 0) in the Sub-TLV field, which is one of the interface parameters of the FEC for the LDP label-mapping message header. This configuration is applicable for all the pseudowires providing full mesh connectivity from the VPLS routing instance to all its neighbors.

To configure the FAT flow label for an FEC 128 VPLS pseudowire, on the ingress PE router:

1. Configure the FEC 128 VPLS routing instance.

```
[edit]
user@PE1# set routing-instances instance-name instance-type vpls
user@PE1# set routing-instances instance-name interface interface-name
user@PE1# set routing-instances instance-name protocols vpls vpls-id vpls-id
user@PE1# set routing-instances instance-name protocols vpls neighbor neighbor-id
user@PE1# set routing-instances instance-name protocols vpls neighbor neighbor-id
```

2. Configure the routing instance to signal the capability to push the flow label in the transmit direction to the remote PE router.

```
[edit routing-instances instance-name protocols vpls]
user@PE1# set flow-label-transmit
```

3. Configure the routing instance to signal the capability to pop the flow label in the receive direction to the remote PE router.

```
[edit routing-instances instance-name protocols vpls]
user@PE1# set flow-label-receive
```

4. Verify and commit the configuration.

   For example:

```
[edit routing-instances]
user@PE1# show
ldp-fec128-signaled-vpls {
    instance-type vpls;
    interface ge-0/0/1.0;
    protocols {
        vpls {
            vpls-id 100;
            flow-label-transmit;
            flow-label-receive;
        }
    }
}
```

5. Repeat the configuration on the remote egress PE router.

RELATED DOCUMENTATION

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | **598**

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | **601**

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic | **772**

FAT Flow Labels Overview | **496**

Configuring the FAT Flow Label for FEC 129 VPLS Pseudowires for Load-Balancing MPLS Traffic

This topic shows how to configure flow-aware transport of pseudowires (FAT) flow labels for forwarding equivalence class (FEC) 129 virtual private LAN service (VPLS) pseudowires for load-balancing MPLS traffic.

Before you begin:

1. Configure the device interfaces and enable MPLS on all the interfaces.
2. Configure MPLS and an LSP from the ingress provider edge (PE) router to the remote PE router.
3. Configure IBGP sessions between the PE routers.
4. Configure an interior gateway protocol (IGP) on the PE router and provider (P) routers.
5. Configure the circuits between the PE routers and the customer edge (CE) routers.
6. Configure LDP on all the interfaces.

FAT flow labels enable load-balancing of MPLS packets across equal-cost multipath (ECMP) paths or link aggregation groups (LAGs) without the need for deep packet inspection of the payload. FAT flow labels can be used for LDP-signaled FEC 128 and FEC 129 pseudowires for VPLS and virtual private wire service (VPWS) networks.

You can configure FAT flow labels to be signaled by LDP on FEC 129 VPLS pseudowires by including the `flow-label-transmit` and `flow-label-receive` configuration statements at the `[edit routing-instances instance-name protocols vpls]` hierarchy level. This configuration sets the T bit and R bit advertisement to 1 (the default being 0) in the Sub-TLV field, which is part one of the interface parameters of the FEC for the LDP label-mapping message header. This configuration is applicable for all the pseudowires providing full mesh connectivity from the VPLS routing instance to all its neighbors.

To configure the FAT flow label for an FEC 129 VPLS pseudowire, on the ingress PE router:

1. Configure the FEC 129 VPLS routing instance.

```
[edit]
user@PE1# set routing-instances instance-name instance-type vpls
user@PE1# set routing-instances instance-name interface interface-name
user@PE1# set routing-instances instance-name route-distinguisher (as-number:id | ip-
address:id)
user@PE1# set routing-instances instance-name l2vpn-id (as-number:x:y | ip-address:id)
user@PE1# set routing-instances instance-name vrf-target target:x:y
```

2. Configure the routing instance to signal the capability to push the flow label in the transmit direction to the remote PE router.

```
[edit routing-instances instance-name protocols vpls]
user@PE1# set flow-label-transmit
```

3. Configure the routing instance to signal the capability to pop the flow label in the receive direction to the remote PE router.

```
[edit routing-instances instance-name protocols vpls]
user@PE1# set flow-label-receive
```

4. Verify and commit the configuration.

   For example:

```
[edit routing-instances]
user@PE1# show
ldp-fec129-signaled-vpls {
    instance-type vpls;
    interface fe-0/0/0.0;
    route-distinguisher 10.255.245.45:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        vpls {
            flow-label-transmit;
            flow-label-receive;
        }
    }
}
```

5. Repeat the configuration on the remote egress PE router.

RELATED DOCUMENTATION

Configuring the FAT Flow Label for FEC 128 VPWS Pseudowires for Load-Balancing MPLS Traffic | **598**

Configuring the FAT Flow Label for FEC 128 VPLS Pseudowires for Load-Balancing MPLS Traffic | **770**

Configuring the FAT Flow Label for FEC 129 VPWS Pseudowires for Load-Balancing MPLS Traffic | **601**

FAT Flow Labels Overview | **496**

Example: Configuring H-VPLS BGP-Based and LDP-Based VPLS Interoperation

IN THIS SECTION

- Requirements | **775**
- Overview | **775**
- Configuration | **776**
- Verification | **793**

This example shows how to configure the hierarchical virtual private LAN service (H-VPLS) in a scenario that uses both LDP-based VPLS and BGP-based VPLS interoperating in a multihoming deployment. This scenario is useful when a customer deployment has the two different types of VPLS in use, and you need to integrate them. Another example is when ISP-A is running BGP-based VPLS and ISP-B is running the LDP-based VPLS, and the two ISPs are merging their networks.

Requirements

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

Overview

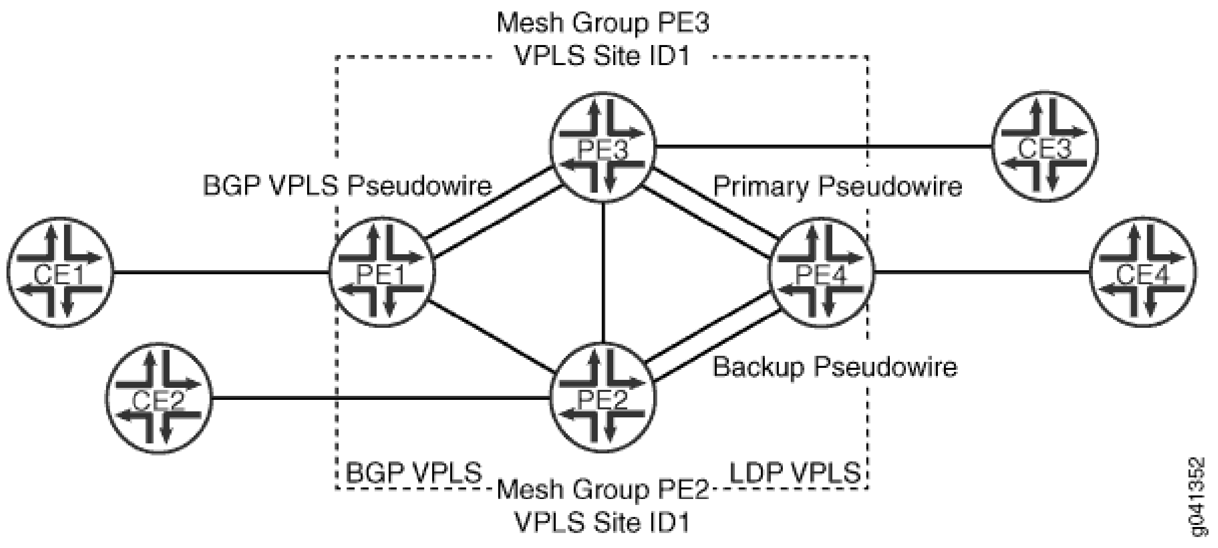
In this example, Device PE2 and Device PE3 are acting as internetworking provider edge (PE) routers with BGP-based as well as LDP-based VPLS termination.

The devices in this example have the following roles:

- BGP VPLS only PE—Device PE1
- LDP VPLS only PE—Device PE4
- BGP-LDP VPLS PE—Device PE2 and Device PE3

Figure 57 on page 775 shows the topology used in this example.

Figure 57: H-VPLS with LDP-Based and BGP-Based VPLS Interoperation



From Device PE4, the pseudowire to Device PE3 is the primary or working path. The pseudowire Device PE2 is the backup path.

"CLI Quick Configuration" on page 776 shows the configuration for all of the devices in Figure 57 on page 775. The section "Step-by-Step Procedure" on page 780 describes the steps on Device PE1, Device PE2, and Device PE4.

Configuration

IN THIS SECTION

- Procedure | 776

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 PE1

```
set interfaces ge-2/0/5 encapsulation ethernet-vpls
set interfaces ge-2/0/5 unit 0 description to_CE1
set interfaces ge-2/0/5 unit 0 family vpls
set interfaces fe-2/0/9 unit 0 description to_PE2
set interfaces fe-2/0/9 unit 0 family inet address 10.10.3.1/30
set interfaces fe-2/0/9 unit 0 family mpls
set interfaces fe-2/0/10 unit 0 description to_PE3
set interfaces fe-2/0/10 unit 0 family inet address 10.10.1.1/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.1/24
set protocols mpls interface fe-2/0/10.0
set protocols mpls interface fe-2/0/9.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface fe-2/0/9.0
set protocols ldp interface lo0.0
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers local-address 192.0.2.1
set protocols bgp group internal-peers family l2vpn signaling
set protocols bgp group internal-peers neighbor 192.0.2.2
set protocols bgp group internal-peers neighbor 192.0.2.3
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface fe-2/0/9.0
set routing-instances h-vpls-PE1 instance-type vpls
set routing-instances h-vpls-PE1 interface ge-2/0/5.0
set routing-instances h-vpls-PE1 route-distinguisher 1:1
set routing-instances h-vpls-PE1 vrf-target target:1:1
set routing-instances h-vpls-PE1 protocols vpls interface ge-2/0/5.0
set routing-instances h-vpls-PE1 protocols vpls site PE1-vpls site-identifier 2
set routing-options autonomous-system 64510
```

Device PE2

```
set interfaces ge-2/0/6 encapsulation ethernet-vpls
set interfaces ge-2/0/6 unit 0 description to_CE2
set interfaces ge-2/0/6 unit 0 family vpls
set interfaces fe-2/0/8 unit 0 description to_PE3
set interfaces fe-2/0/8 unit 0 family inet address 10.10.4.2/30
set interfaces fe-2/0/8 unit 0 family mpls
set interfaces fe-2/0/9 unit 0 description to_PE4
set interfaces fe-2/0/9 unit 0 family inet address 10.10.5.1/30
set interfaces fe-2/0/9 unit 0 family mpls
set interfaces fe-2/0/10 unit 0 description to_PE1
set interfaces fe-2/0/10 unit 0 family inet address 10.10.3.2/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.2/24
set protocols mpls interface fe-2/0/10.0
set protocols mpls interface fe-2/0/9.0
set protocols mpls interface fe-2/0/8.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface fe-2/0/9.0
set protocols ldp interface fe-2/0/8.0
set protocols ldp interface lo0.0
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.0.2.2
set protocols bgp group ibgp family l2vpn signaling
set protocols bgp group ibgp neighbor 192.0.2.3
set protocols bgp group ibgp neighbor 192.0.2.1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface fe-2/0/9.0
set protocols ospf area 0.0.0.0 interface fe-2/0/8.0
set routing-instances h-vpls-PE2 instance-type vpls
set routing-instances h-vpls-PE2 interface ge-2/0/6.0
set routing-instances h-vpls-PE2 route-distinguisher 1:2
set routing-instances h-vpls-PE2 vrf-target target:1:1
set routing-instances h-vpls-PE2 protocols vpls interface ge-2/0/6.0
set routing-instances h-vpls-PE2 protocols vpls site PE2-vpls site-identifier 1
set routing-instances h-vpls-PE2 protocols vpls site PE2-vpls multi-homing
set routing-instances h-vpls-PE2 protocols vpls site PE2-vpls mesh-group h-vpls-PE2
set routing-instances h-vpls-PE2 protocols vpls vpls-id 100
set routing-instances h-vpls-PE2 protocols vpls mesh-group h-vpls-PE2 vpls-id 100
set routing-instances h-vpls-PE2 protocols vpls mesh-group h-vpls-PE2 local-switching
set routing-instances h-vpls-PE2 protocols vpls mesh-group h-vpls-PE2 neighbor 192.0.2.4
set routing-options autonomous-system 64510
```

Device PE3

```
set interfaces fe-2/0/8 unit 0 description to_PE2
set interfaces fe-2/0/8 unit 0 family inet address 10.10.4.1/30
set interfaces fe-2/0/8 unit 0 family mpls
set interfaces fe-2/0/9 unit 0 description to_PE4
set interfaces fe-2/0/9 unit 0 family inet address 10.10.6.1/30
set interfaces fe-2/0/9 unit 0 family mpls
set interfaces fe-2/0/10 unit 0 description to_PE1
set interfaces fe-2/0/10 unit 0 family inet address 10.10.1.2/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces ge-2/1/3 encapsulation ethernet-vpls
set interfaces ge-2/1/3 unit 0 description to_CE3
set interfaces ge-2/1/3 unit 0 family vpls
set interfaces lo0 unit 0 family inet address 192.0.2.3/24
set protocols mpls interface fe-2/0/10.0
set protocols mpls interface fe-2/0/8.0
set protocols mpls interface fe-2/0/9.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface fe-2/0/9.0
set protocols ldp interface fe-2/0/8.0
set protocols ldp interface lo0.0
set protocols bgp group internal-peers type internal
set protocols bgp group internal-peers local-address 192.0.2.3
set protocols bgp group internal-peers family l2vpn signaling
set protocols bgp group internal-peers neighbor 192.0.2.2
set protocols bgp group internal-peers neighbor 192.0.2.1
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface fe-2/0/8.0
set protocols ospf area 0.0.0.0 interface fe-2/0/9.0
set routing-instances h-vpls-PE3 instance-type vpls
set routing-instances h-vpls-PE3 interface ge-2/1/3.0
set routing-instances h-vpls-PE3 route-distinguisher 1:3
set routing-instances h-vpls-PE3 vrf-target target:1:1
set routing-instances h-vpls-PE3 protocols vpls interface ge-2/1/3.0
set routing-instances h-vpls-PE3 protocols vpls site PE3-vpls site-identifier 1
set routing-instances h-vpls-PE3 protocols vpls site PE3-vpls multi-homing
set routing-instances h-vpls-PE3 protocols vpls site PE3-vpls mesh-group h-vpls-PE3
set routing-instances h-vpls-PE3 protocols vpls vpls-id 100
set routing-instances h-vpls-PE3 protocols vpls mesh-group h-vpls-PE3 vpls-id 100
set routing-instances h-vpls-PE3 protocols vpls mesh-group h-vpls-PE3 local-switching
set routing-instances h-vpls-PE3 protocols vpls mesh-group h-vpls-PE3 neighbor 192.0.2.4
set routing-options autonomous-system 64510
```

Device PE4

```
set interfaces fe-2/0/9 unit 0 description to_PE3
set interfaces fe-2/0/9 unit 0 family inet address 10.10.6.2/30
set interfaces fe-2/0/9 unit 0 family mpls
set interfaces fe-2/0/10 unit 0 description to_PE2
set interfaces fe-2/0/10 unit 0 family inet address 10.10.5.2/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces ge-2/1/7 encapsulation ethernet-vpls
set interfaces ge-2/1/7 unit 0 description to_CE4
set interfaces ge-2/1/7 unit 0 family vpls
set interfaces lo0 unit 0 family inet address 192.0.2.4/24
set protocols mpls interface fe-2/0/10.0
set protocols mpls interface fe-2/0/9.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface fe-2/0/9.0
set protocols ldp interface lo0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface fe-2/0/9.0
set routing-instances ldp-vpls instance-type vpls
set routing-instances ldp-vpls interface ge-2/1/7.0
set routing-instances ldp-vpls protocols vpls vpls-id 100
set routing-instances ldp-vpls protocols vpls neighbor 192.0.2.3
set routing-instances ldp-vpls protocols vpls neighbor 192.0.2.2
```

Device CE1

```
set interfaces ge-2/0/8 unit 0 description to_PE1
set interfaces ge-2/0/8 unit 0 family inet address 172.16.0.1/24
set interfaces lo0 unit 0 family inet address 10.255.14.214/32
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 CE2

```
set interfaces ge-2/1/5 unit 0 description to_PE2
set interfaces ge-2/1/5 unit 0 family inet address 172.16.0.2/24
set interfaces lo0 unit 0 family inet address 10.255.14.215/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/5.0
```

Device CE3

```
set interfaces ge-2/0/9 unit 0 description to_PE3
set interfaces ge-2/0/9 unit 0 family inet address 172.16.0.3/24
set interfaces lo0 unit 0 family inet address 10.255.14.218/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/9.0
```

Device CE4

```
set interfaces ge-2/1/6 unit 0 description to_PE4
set interfaces ge-2/1/6 unit 0 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 10.255.14.219/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/6.0
```

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 CLI User Guide.

To configure the BGP-based VPLS PE device:

1. Configure the interfaces.

   On the device interface that connects to the customer edge, enable VPLS encapsulation and the VPLS address family.

   On the core-facing interfaces, enable MPLS labels.

   ```
   [edit interfaces]
   user@PE1# set ge-2/0/5 encapsulation ethernet-vpls
   user@PE1# set ge-2/0/5 unit 0 description to_CE1
   user@PE1# set ge-2/0/5 unit 0 family vpls
   user@PE1# set fe-2/0/10 unit 0 description to_PE3
   user@PE1# set fe-2/0/10 unit 0 family inet address 10.10.1.1/30
   user@PE1# set fe-2/0/10 unit 0 family mpls
   user@PE1# set fe-2/0/9 unit 0 description to_PE2
   user@PE1# set fe-2/0/9 unit 0 family inet address 10.10.3.1/30
   user@PE1# set fe-2/0/9 unit 0 family mpls
   user@PE1# set lo0 unit 0 family inet address 192.0.2.1/24
   ```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

   ```
   [edit protocols mpls]
   user@PE1# set interface fe-2/0/10.0
   user@PE1# set interface fe-2/0/9.0
   [edit protocols ldp ]
   user@PE1# set interface fe-2/0/10.0
   user@PE1# set interface fe-2/0/9.0
   user@PE1# set interface lo0.0
   ```

3. Enable routing on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

   ```
   [edit protocols ospf area 0.0.0.0]
   user@PE1# set interface lo0.0 passive
   user@PE1# set interface fe-2/0/10.0
   user@PE1# set interface fe-2/0/9.0
   ```

4. Configure BGP with Layer 2 VPN signaling.

   The `l2vpn signaling` statement enables support for both VPLS and Layer 2 VPN advertisement under the same network layer reachability information (NLRI).

   The internal IBGP (IBGP) full mesh includes Device PE1, Device PE2, and Device PE3. Device PE4 is not included.

```
[edit protocols bgp group internal-peers]
user@PE1# set type internal
user@PE1# set local-address 192.0.2.1
user@PE1# set family l2vpn signaling
user@PE1# set neighbor 192.0.2.2
user@PE1# set neighbor 192.0.2.3
```

5. Configure the VPLS routing instance.

   Because this is BGP-based VPLS, include a route distinguisher, a VRF target, and a site name and ID.

```
[edit routing-instances h-vpls-PE1]
user@PE1# set instance-type vpls
user@PE1# set interface ge-2/0/5.0
user@PE1# set route-distinguisher 1:1
user@PE1# set vrf-target target:1:1
[edit routing-instances h-vpls-PE1 protocols vpls]
user@PE1# set interface ge-2/0/5.0
user@PE1# set site PE1-vpls site-identifier 2
```

6. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@PE1# set autonomous-system 64510
```

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 BGP-LDP-based VPLS PE device:

1. Configure the interfaces.

   On the PE-r device interface that connects to the customer edge, configure one of the VPLS encapsulation types and the VPLS address family. This enables VPLS.

   On the core-facing interfaces, enable MPLS labels.

   ```
   [edit interfaces]
   user@PE2# set ge-2/0/6 encapsulation ethernet-vpls
   user@PE2# set ge-2/0/6 unit 0 description to_CE2
   user@PE2# set ge-2/0/6 unit 0 family vpls
   user@PE2# set fe-2/0/10 unit 0 description to_PE1
   user@PE2# set fe-2/0/10 unit 0 family inet address 10.10.3.2/30
   user@PE2# set fe-2/0/10 unit 0 family mpls
   user@PE2# set fe-2/0/9 unit 0 description to_PE4
   user@PE2# set fe-2/0/9 unit 0 family inet address 10.10.5.1/30
   user@PE2# set fe-2/0/9 unit 0 family mpls
   user@PE2# set fe-2/0/8 unit 0 description to_PE3
   user@PE2# set fe-2/0/8 unit 0 family inet address 10.10.4.2/30
   user@PE2# set fe-2/0/8 unit 0 family mpls
   user@PE2# set lo0 unit 0 family inet address 192.0.2.2/24
   ```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

   ```
   [edit protocols mpls]
   user@PE2# set interface fe-2/0/10.0
   user@PE2# set interface fe-2/0/9.0
   user@PE2# set interface fe-2/0/8.0
   [edit protocols ldp]
   user@PE2# set interface fe-2/0/10.0
   user@PE2# set interface fe-2/0/9.0
   user@PE2# set interface fe-2/0/8.0
   user@PE2# set interface lo0.0
   ```

3. Enable routing on the interfaces.

On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

```
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface lo0.0 passive
user@PE2# set interface fe-2/0/10.0
user@PE2# set interface fe-2/0/9.0
user@PE2# set interface fe-2/0/8.0
```

4. Configure BGP with Layer 2 VPN signaling.

   The l2vpn signaling statement enables support for both VPLS and Layer 2 VPN advertisement under the same network layer reachability information (NLRI)

   The internal IBGP (IBGP) full mesh includes Device PE1, Device PE2, and Device PE3. Device PE4 is not included.

```
[edit protocols bgp group ibgp]
user@PE2# set type internal
user@PE2# set local-address 192.0.2.2
user@PE2# set family l2vpn signaling
user@PE2# set neighbor 192.0.2.3
user@PE2# set neighbor 192.0.2.1
```

5. Configure VPLS.

   The `vpls-id` statement enables LDP signaling for the VPLS instance.

```
[edit routing-instances h-vpls-PE2]
user@PE2# set instance-type vpls
user@PE2# set interface ge-2/0/6.0
user@PE2# set route-distinguisher 1:2
user@PE2# set vrf-target target:1:1
[edit routing-instances h-vpls-PE2 protocols vpls]
user@PE2# set interface ge-2/0/6.0
user@PE2# set site PE2-vpls site-identifier 1
user@PE2# set site PE2-vpls multi-homing
user@PE2# set site PE2-vpls mesh-group h-vpls-PE2
user@PE2# set vpls-id 100
user@PE2# set mesh-group h-vpls-PE2 vpls-id 100
```

```
user@PE2# set mesh-group h-vpls-PE2 local-switching
user@PE2# set mesh-group h-vpls-PE2 neighbor 192.0.2.4
```

6. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@PE2# set autonomous-system 64510
```

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 LDP-based VPLS PE device:

1. Configure the interfaces.

   On the PE-r device interface that connects to the customer edge, configure one of the VPLS encapsulation types and the VPLS address family. This enables VPLS.

   On the core-facing interfaces, enable MPLS labels.

```
[edit interfaces]
user@PE4# set fe-2/0/10 unit 0 description to_PE2
user@PE4# set fe-2/0/10 unit 0 family inet address 10.10.5.2/30
user@PE4# set fe-2/0/10 unit 0 family mpls
user@PE4# set fe-2/0/9 unit 0 description to_PE3
user@PE4# set fe-2/0/9 unit 0 family inet address 10.10.6.2/30
user@PE4# set fe-2/0/9 unit 0 family mpls
user@PE4# set ge-2/1/7 encapsulation ethernet-vpls
user@PE4# set ge-2/1/7 unit 0 description to_CE4
user@PE4# set ge-2/1/7 unit 0 family vpls
user@PE4# set lo0 unit 0 family inet address 192.0.2.4/24
```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

```
[edit protocols mpls]
user@PE4# set interface fe-2/0/10.0
```

```
user@PE4# set interface fe-2/0/9.0
[edit protocols ldp]
user@PE4# set interface fe-2/0/10.0
user@PE4# set interface fe-2/0/9.0
user@PE4# set interface lo0.0
```

3. Enable routing on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

```
[edit protocols ospf area 0.0.0.0]
user@PE4# set interface lo0.0 passive
user@PE4# set interface fe-2/0/10.0
user@PE4# set interface fe-2/0/9.0
```

4. Configure VPLS.

   The `vpls-id` statement enables LDP signaling for the VPLS instance.

```
[edit routing-instances ldp-vpls]
user@PE4# set instance-type vpls
user@PE4# set interface ge-2/1/7.0
user@PE4# set protocols vpls vpls-id 100
[edit routing-instances ldp-vpls protocols vpls]
user@PE4# set neighbor 192.0.2.3
user@PE4# set neighbor 192.0.2.2
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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.

Device PE1

```
user@PE1# show interfaces
ge-2/0/5 {
    encapsulation ethernet-vpls;
    unit 0 {
        description to_CE1;
        family vpls;
    }
}
fe-2/0/9 {
}
    unit 0 {
        description to_PE2;
        family inet {
            address 10.10.3.1/30;
        }
        family mpls;
    }
}
fe-2/0/10 {
    unit 0 {
        description to_PE3;
        family inet {
            address 10.10.1.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface fe-2/0/10.0;
    interface fe-2/0/9.0;
}
bgp {
    group internal-peers {
        type internal;
        local-address 192.0.2.1;
        family l2vpn {
            signaling;
        }
        neighbor 192.0.2.2;
        neighbor 192.0.2.3;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-2/0/10.0;
        interface fe-2/0/9.0;
    }
}
ldp {
    interface fe-2/0/10.0;
    interface fe-2/0/9.0;
    interface lo0.0;
}
```

```
user@PE1# show routing-instances
h-vpls-PE1 {
    instance-type vpls;
    interface ge-2/0/5.0;
    route-distinguisher 1:1;
    vrf-target target:1:1;
    protocols {
        vpls {
            interface ge-2/0/5.0;
            site PE1-vpls {
                site-identifier 2;
            }
        }
    }
}
```

```
user@PE1# show routing-options
autonomous-system 64510;
```

Device PE2

```
user@PE2# show interfaces
ge-2/0/6 {
    encapsulation ethernet-vpls;
    unit 0 {
        description to_CE2;
        family vpls;
    }
}
fe-2/0/8 {
    unit 0 {
        description to_PE3;
        family inet {
            address 10.10.4.2/30;
        }
        family mpls;
    }
}
fe-2/0/9 {
    unit 0{
        description to_PE4;
        family inet {
            address 10.10.5.1/30;
        }
        family mpls;
    }
}
fe-2/0/10 {
    unit 0 {
        description to_PE1;
        family inet {
            address 10.10.3.2/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.2/24;
        }
    }
}
```

```
user@PE2# show protocols
mpls {
    interface fe-2/0/10.0;
    interface fe-2/0/9.0;
    interface fe-2/0/8.0;
}
bgp {
    group ibgp {
        type internal;
        local-address 192.0.2.2;
        family l2vpn {
            signaling;
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-2/0/10.0;
        interface fe-2/0/9.0;
        interface fe-2/0/8.0;
    }
}
ldp {
    interface fe-2/0/10.0;
    interface fe-2/0/9.0;
    interface fe-2/0/8.0;
    interface lo0.0;
}
```

```
user@PE2# show routing-instances
h-vpls-PE2 {
    instance-type vpls;
    interface ge-2/0/6.0;
    route-distinguisher 1:2;
    vrf-target target:1:1;
    protocols {
        vpls {
            interface ge-2/0/6.0;
            site PE2-vpls {
                site-identifier 1;
                multi-homing;
                mesh-group h-vpls-PE2;
            }
            vpls-id 100;
            mesh-group h-vpls-PE2 {
                vpls-id 100;
                local-switching;
                neighbor 192.0.2.4;
            }
        }
    }
}
```

```
user@PE2# show routing-options
autonomous-system 64510;
```

Device PE4

```
user@PE4# show interfaces
ge-2/1/7 {
    encapsulation ethernet-vpls;
    unit 0 {
        description to_CE4;
        family vpls;
    }
}
fe-2/0/9 {
    unit 0 {
        description to_PE3;
        family inet {
            address 10.10.6.2/30;
        }
        family mpls;
    }
}
fe-2/0/10 {
    unit 0 {
        description to_PE2;
        family inet {
            address 10.10.5.2/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.4/24;
        }
    }
}
```

```
user@PE4# show protocols
mpls {
    interface fe-2/0/9.0;
    interface fe-2/0/10.0;
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-2/0/9.0;
        interface fe-2/0/10.0;
    }
}
ldp {
    interface fe-2/0/10.0;
    interface fe-2/0/9.0;
    interface lo0.0;
}
```

```
user@PE4# show routing-instances
ldp-vpls {
    instance-type vpls;
    interface ge-2/1/7.0;
    protocols {
        vpls {
            vpls-id 100;
            neighbor 192.0.2.3;
            neighbor 192.0.2.2;
        }
    }
}
```

If you are done configuring the devices, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying the VPLS Connections | **793**
- Manually Triggering a Switch from the Active Pseudowire to the Backup Pseudowire | **798**
- Checking Connectivity | **803**
- Checking the BGP Layer 2 VPN Routing Tables | **803**
- Checking the Layer 2 Circuit Routing Tables | **804**

Confirm that the configuration is working properly. In a multihoming scenario with BGP-LDP VPLS, the LDP pseudowires are in the down state for the backup PE (Device PE2). Whereas on the LDP-only VPLS PE (Device PE4), the pseudowires to the primary and backup BGP-LDP PE devices are in the up state.

Verifying the VPLS Connections

Purpose

Verify that the VPLS connections are working as expected.

Action

From operational mode, enter the `show vpls connections` command.

```
user@PE1> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE1
  Local site: PE1-vpls (2)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Oct 16 16:52:27 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800016, Outgoing label: 800009
      Local interface: vt-2/0/10.51380738, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls h-vpls-PE1 local site 2 remote site 1
```

```
user@PE2> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE2
  BGP-VPLS State
  Local site: PE2-vpls (1)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   RN
    2                         rmt   Up     Oct 16 17:12:31 2012           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800257, Outgoing label: 800000
      Local interface: vt-2/0/10.118489089, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls h-vpls-PE2 local site 1 remote site 2
  LDP-VPLS State
  VPLS-id: 100
  Mesh-group connections: h-vpls-PE2
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.4(vpls-id 100)      rmt   Up     Oct 16 17:12:30 2012           1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800001
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.118489088, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls h-vpls-PE2 neighbor 192.0.2.4 vpls-id 100
```

```
user@PE3> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE3
  BGP-VPLS State
  Local site: PE3-vpls (1)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   LN
    2                         rmt   LN
  LDP-VPLS State
  VPLS-id: 100
  Mesh-group connections: h-vpls-PE3
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.4(vpls-id 100)      rmt   LN
```

```
user@PE4> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: ldp-vpls
  VPLS-id: 100
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.2(vpls-id 100)      rmt   Up     Oct 16 17:12:23 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 800000
```

```
        Negotiated PW status TLV: No
        Local interface: vt-2/0/10.17825793, Status: Up, Encapsulation: ETHERNET
          Description: Intf - vpls ldp-vpls neighbor 192.0.2.2 vpls-id 100
      192.0.2.3(vpls-id 100)      rmt   Up     Oct 16 17:12:20 2012           1
        Remote PE: 192.0.2.3, Negotiated control-word: No
        Incoming label: 800000, Outgoing label: 800000
        Negotiated PW status TLV: No
        Local interface: vt-2/0/10.17825792, Status: Up, Encapsulation: ETHERNET
          Description: Intf - vpls ldp-vpls neighbor 192.0.2.3 vpls-id 100
```

Meaning

On Device PE1, the BGP-VPLS connection to Device PE2 is up. In a steady-state condition, Device PE2 is the primary router and has all pseudowires terminating on it. Traffic flows from CE1 to PE1 to PE2 to PE4 to CE4.

On Device PE2, the BGP-VPLS connection to Device PE1 is up. The connection to Device PE3 is in the RN state. The LDP-VPLS connection to Device PE4 is up.

On Device PE3, all VPLS connections are in the LN state. This is expected because Device PE3 is the backup.

On Device PE4, the LDP-only VPLS router, the primary pseudowire to Device PE2 and the backup pseudowire to Device PE3 are in the up state.

Manually Triggering a Switch from the Active Pseudowire to the Backup Pseudowire

Purpose

Verify that when Device PE2 becomes unavailable, the traffic flow shifts to Device PE3.

Action

1. On Device PE2, deactivate the interfaces.

   ```
   user@PE2# deactivate interfaces
   user@PE2# commit
   ```

2. Rerun the `show vpls connections` command on all of the PE devices.

```
user@PE1> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE1
  Local site: PE1-vpls (2)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Oct 17 12:24:01 2012           2
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800257
      Local interface: vt-2/0/10.84934656, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls h-vpls-PE1 local site 2 remote site 1
```

```
user@PE2> show vpls connections
Layer-2 VPN connections:
```

```
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE2
  BGP-VPLS State
  Local site: PE2-vpls (1)
  LDP-VPLS State
  VPLS-id: 100
  Mesh-group connections: h-vpls-PE2
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.4(vpls-id 100)      rmt   OL
```

```
user@PE3> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
```

```
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: h-vpls-PE3
  BGP-VPLS State
  Local site: PE3-vpls (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 17 12:24:01 2012           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800257, Outgoing label: 800000
      Local interface: vt-2/0/10.135266304, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls h-vpls-PE3 local site 1 remote site 2
  LDP-VPLS State
  VPLS-id: 100
  Mesh-group connections: h-vpls-PE3
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.4(vpls-id 100)      rmt   Up     Oct 17 12:24:02 2012           1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800000
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.135266305, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls h-vpls-PE3 neighbor 192.0.2.4 vpls-id 100
```

```
user@PE4> show vpls connections
Layer-2 VPN connections:
```

```
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: ldp-vpls
  VPLS-id: 100
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.2(vpls-id 100)      rmt   OL
    192.0.2.3(vpls-id 100)      rmt   Up     Oct 16 17:12:20 2012           1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800000
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.17825792, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls ldp-vpls neighbor 192.0.2.3 vpls-id 100
```

Meaning

On Device PE1, the BGP-VPLS connection to Device PE3 is up. Traffic flows from CE1 to PE1 to PE3 to PE4 to CE4.

On Device PE2, the BGP-VPLS connection to Device PE1 is in the OL state.

On Device PE3, all VPLS connections are up.

On Device PE4, the VPLS connection to Device PE2 is in the OL state. The VPLS connection to Device PE3 is up.

If you reactivate the interfaces on Device PE2, the connections revert to their previous state and traffic flow.

Checking Connectivity

Purpose

Verify that Device CE1 can ping Device CE4.

Action

```
user@CE1> ping 10.255.14.219

PING 10.255.14.219 (10.255.14.219): 56 data bytes
64 bytes from 10.255.14.219: icmp_seq=0 ttl=64 time=1.149 ms
64 bytes from 10.255.14.219: icmp_seq=1 ttl=64 time=0.779 ms
^C
--- 10.255.14.219 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.779/0.964/1.149/0.185 ms
```

Meaning

The output shows that VPLS is operational.

Checking the BGP Layer 2 VPN Routing Tables

Purpose

Verify that the VPLS routes are learned from BGP.

Action

```
user@PE1> show route table bgp.l2vpn.0
bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
```

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

1:3:1:1/96
                   *[BGP/170] 20:00:11, localpref 100, from 192.0.2.3
                      AS path: I, validation-state: unverified
                    > to 10.10.1.2 via fe-2/0/10.0

user@PE3> show route table bgp.l2vpn.0
bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:1:2:1/96
                   *[BGP/170] 20:00:11, localpref 100, from 192.0.2.1
                      AS path: I, validation-state: unverified
                    > to 10.10.1.1 via fe-2/0/10.0
```

Checking the Layer 2 Circuit Routing Tables

Purpose

Verify that the VPLS routes are learned from LDP.

Action

```
user@PE3> show route table l2circuit.0
l2circuit.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.4:NoCtrlWord:5:100:Local/96
                   *[VPLS/7] 01:30:11, metric2 1
                    > to 10.10.6.2 via fe-2/0/9.0
192.0.2.4:NoCtrlWord:5:100:Remote/96
                   *[LDP/9] 20:41:57
                      Discard
user@PE4> show route table bgp.l2vpn.0
l2circuit.0: 3 destinations, 3 routes (2 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:NoCtrlWord:5:100:Local/96
                   *[VPLS/7] 20:42:51, metric2 1
                    > to 10.10.6.1 via fe-2/0/9.0
```

```
192.0.2.3:NoCtrlWord:5:100:Remote/96
                   *[LDP/9] 20:41:57
                      Discard
```

RELATED DOCUMENTATION

Application Note: Demystifying H-VPLS

Example: Configuring H-VPLS Without VLANs | 860

Example: Configuring H-VPLS With VLANs | 842

Redundant Pseudowires for Layer 2 Circuits and VPLS | 224

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | 226

Example: Configuring BGP-Based H-VPLS Using Different Mesh Groups for Each Spoke Router

IN THIS SECTION

- Requirements | 805
- Overview and Topology | 806
- Configuration | 808

This example shows how to configure the hierarchical virtual private LAN service (H-VPLS) using different mesh groups to provide H-VPLS functionality and provides steps for verifying the configuration. This is one type of H-VPLS configuration possible in the Juniper Networks implementation. For information about the alternate type of configuration see "Example: Configuring LDP-Based H-VPLS Using a Single Mesh Group to Terminate the Layer 2 Circuits" on page 833.

Using mesh groups improves LDP-based VPLS control plane scalability and avoids the requirement for a full mesh of LDP sessions. This example uses BGP-based VPLS.

This example is organized into the following sections:

Requirements

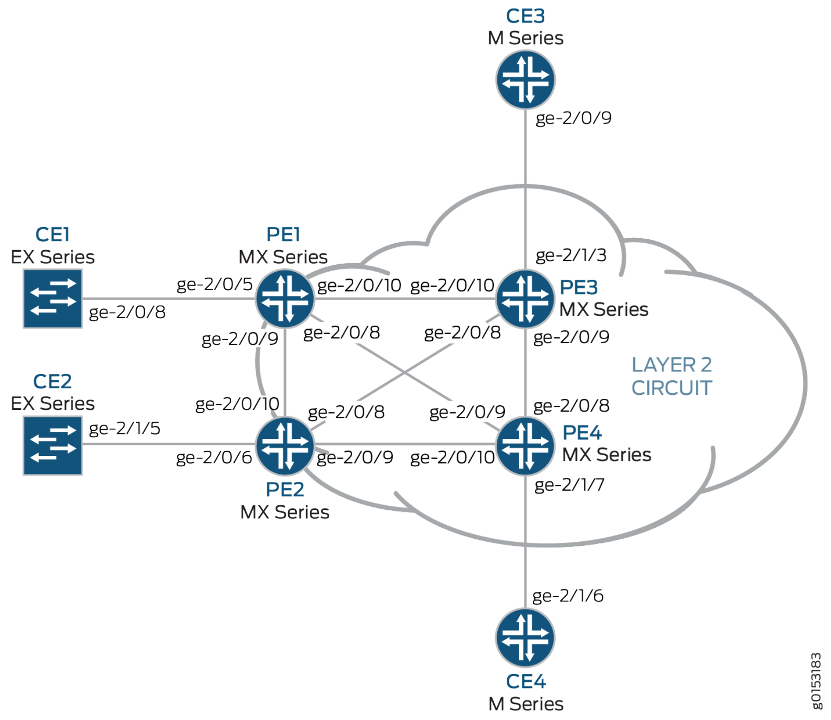
This example uses the following hardware components:

- Four MX Series 5G Universal Routing Platforms for Router PE1, Router PE2, Router PE3, and Router PE4
- One M Series Multiservice Edge Router for Router CE4
- Two EX Series Ethernet Switches for Device CE1 and Device CE2
- One J Series Services Router for Router CE3

Overview and Topology

Figure 58 on page 806 shows the physical topology used in this example.

Figure 58: Physical Topology of H-VPLS



The following describes the base configuration used in this example:

- Router PE1 and Router PE2 are configured as MTU devices.
- Router PE3 and Router PE4 are configured as PE-r routers, each using an LDP-based VPLS routing instance.
- The LDP and OSPF protocols are configured on all of the MTU devices and PE-r routers.
- Core-facing interfaces are enabled with the MPLS address family.

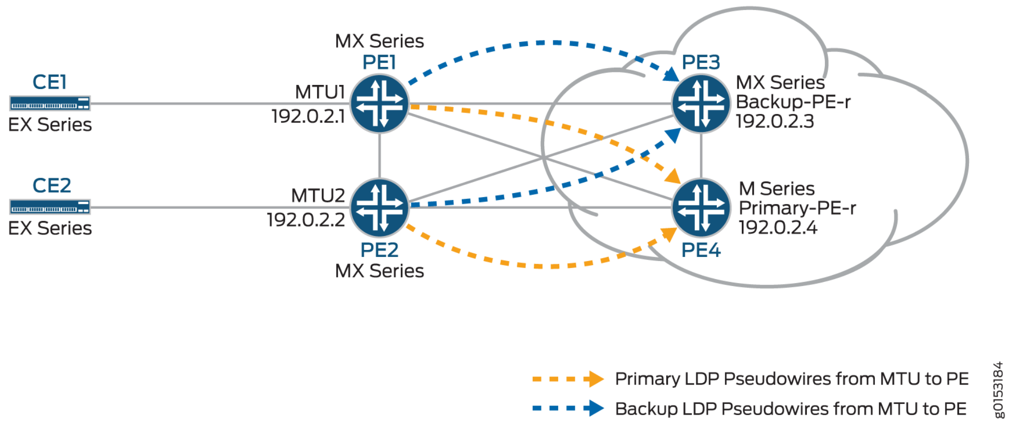
- Optionally, the VPLS routing instances can be configured on PE-r routers with the `no-tunnel-interface` statement. This allows the routers to use a label-switched interface (LSI), which is useful if your routers do not have Tunnel Services PICs or built-in support for tunnel services.

- All of the routers are configured with loopback IP addresses.

- BGP is configured on the PE-r routers. Optionally, you can configure route reflection. This is useful for scaling internal BGP (IBGP). The BGP configuration includes the `signaling` statement at the `[edit protocols bgp group` *`group-name`* `family l2vpn]` hierarchy level to support Layer 2 VPN signaling using BGP.

Figure 59 on page 807 shows the logical topology used in this example.

Figure 59: Logical Topology of H-VPLS



In Figure 59 on page 807:

- The MTU devices (Router PE1 and Router PE2) have Layer 2 circuit connections to the PE-r routers (Router PE3 and Router PE4). For redundancy, a backup neighbor is configured for the Layer 2 circuit connections to the PE-r routers.

- The `l2circuit` statement in the `[edit protocols]` hierarchy is included on the MTU devices.

- A VPLS routing instance is configured on the PE-r routers.

- In the VPLS routing instance on the PE-r routers, mesh groups are created to terminate the Layer 2 circuit pseudowires that originate at the MTU devices.

- Each MTU device is configured with a different virtual circuit ID.

- Each PE-r router's mesh groups configuration includes VPLS ID values that match the virtual circuit IDs used on the MTU devices.

Configuration

IN THIS SECTION

- Configuring the Spoke MTU PE Routers | 808
- Configuring the Hub PE (PE-r) | 810
- Verifying the H-VPLS Operation | 813

To configure H-VPLS with different mesh groups for each spoke PE-r router using BGP-based VPLS, perform the following tasks:

Configuring the Spoke MTU PE Routers

Step-by-Step Procedure

1. On Router PE1, configure the Gigabit Ethernet interface connected to Router CE1. Include the `encapsulation` statement and specify the `ethernet-ccc` option. Also configure the logical interface by including the `family` statement and specifying the `ccc` option.

```
[edit interfaces]
ge-2/0/5 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
```

2. On Router PE1, configure the Layer 2 circuit by including the `neighbor` statement and specifying the IP address of Router PE3 as the neighbor. Configure the Gigabit Ethernet logical interface by including the `virtual-circuit-id` statement and specifying `100` as the ID. Also configure a backup neighbor for the Layer 2 circuit by including the `backup-neighbor` statement, specifying the loopback interface IP address of Router PE4 as the backup neighbor, and including the `standby` statement.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.3 {
        interface ge-2/0/5.0 {
            virtual-circuit-id 100;
            backup-neighbor 192.0.2.4 {          # Backup H-VPLS PE router
            standby;
        }
    }
}
```

3. On Router PE2, configure the Gigabit Ethernet interface connected to Router CE2. Include the `encapsulation` statement and specify the `ethernet-ccc` option. Also configure the logical interface by including the `family` statement and specifying the `ccc` option.

```
[edit interfaces]
ge-2/0/6 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
```

4. On Router PE2, configure the Layer 2 circuit by including the `neighbor` statement and specifying the IP address of Router PE3 as the neighbor. Configure the Gigabit Ethernet logical interface by including the `virtual-circuit-id` statement and specifying `200` as the ID. Configure the encapsulation by including the `encapsulation-type` statement and specifying the `ethernet` option. Also configure a backup neighbor for the Layer 2 circuit by including the `backup-neighbor` statement, specifying the loopback interface IP address of Router PE4 as the backup neighbor, and including the `standby` statement.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.3 {
        interface ge-1/0/2.0 {
            virtual-circuit-id 200;          # different VC-ID
            encapsulation-type ethernet;          # default encapsulation
            backup-neighbor 192.0.2.4 {
                standby;
            }
        }
    }
}
```

Configuring the Hub PE (PE-r)

Step-by-Step Procedure

1. On Router PE3 (the primary hub), configure the Gigabit Ethernet interface connected to Router CE3. Include the `encapsulation` statement and specify the `ethernet-vpls` option. Also configure the logical interface by including the `family vpls` statement.

```
[edit interfaces]
ge-2/0/0 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
    }
}
```

2. On Router PE4 (the backup hub), configure the Gigabit Ethernet interface connected to Router CE4. Include the `encapsulation` statement and specify the `ethernet-vpls` option. Also configure the logical interface by including the `family vpls` statement.

```
[edit interfaces]
ge-2/1/7 {
    encapsulation ethernet-vpls;
    unit 0 {
        description to_CE4;
        family vpls;
    }
}
```

```
lo0 {
    unit 0 {
        family inet {
                address 192.0.2.4/24;
            }
        }
    }
```

3. On PE-r Router PE3, configure the BGP-based VPLS routing instance by including the `instance-type` statement at the `[edit routing-instances H-VPLS]` hierarchy level and specifying the `vpls` option. Include the `interface` statement and specify the Gigabit Ethernet interface connected to Router CE3. Configure a route distinguisher to ensure that the route advertisement is unique by including the `route-distinguisher` statement and specifying `192.0.2.3:33` as the value. Also configure the VPN routing and forwarding (VRF) route target to be included in the route advertisements to the other routers participating in the VPLS. To configure the VRF route target, include the `vrf-target` statement and specify `target:64510:2` as the value. Optionally, include the `no-tunnel-services` statement to enable the use of LSI interfaces, which is useful if the device does not have tunnel services. The `no-tunnel-services` statement is omitted in this example. Optionally, you can include the `site-range` statement to specify an upper limit on the maximum site identifier that can be accepted to allow a pseudowire to be brought up. The `site-range` statement is omitted in this example. We recommend using the default of 65,534.

   Configure the VPLS protocol and the mesh groups for each MTU PE device.

   To configure the VPLS protocol, include the `vpls` statement at the `[edit routing-instances H-VPLS protocols]` hierarchy level. Include the `site` statement and specify a name for the site. Include the `interface` statement and specify the Gigabit Ethernet interface connected to Device CE3.

   Configuring mesh groups under the VPLS instance terminates the Layer 2 circuit into the VPLS instance. To configure each mesh group, include the `mesh-group` statement and specify the mesh group name. In this example, the mesh group name is the name of the MTU device associated with each mesh group. Include the `vpls-id` statement and specify the ID that matches the virtual circuit ID configured in "Configuring the Spoke MTU PE Routers" on page 808. Also include the `neighbor` statement and specify the IP address of the spoke PE router associated with each mesh group. Optionally, include the `local-switching` statement if you are not using a full mesh of VPLS connections. The `local-switching` statement is useful if you are configuring a single mesh group and terminating multiple Layer 2 circuit pseudowires into it. The `local-switching` statement is omitted in this example.

```
routing-instances {
    H-VPLS {
        instance-type vpls;
        interface ge-2/1/3.0;
        route-distinguisher 192.0.2.3:33;
        vrf-target target:64510:2;
        protocols {
            vpls {
                site pe3 {
                    site-identifier 3;
                    interface ge-2/1/3.0;
                }
                mesh-group pe1 {
                    vpls-id 100;
                    neighbor 192.0.2.1;
                }
                mesh-group pe2 {
                    vpls-id 200;
                    neighbor 192.0.2.2;
                }
            }
        }
    }
}
```

4. On PE-r Router PE4, configure a routing instance like the one on Router PE3.

```
routing-instances {
    H-VPLS {
        instance-type vpls;
        interface ge-2/1/7.0;
        route-distinguisher 192.0.2.4:44;
        vrf-target target:64510:2;
        protocols {
            vpls {
                site pe4 {
                    site-identifier 4;
                    interface ge-2/1/7.0;
                }
                mesh-group pe1 {
                    vpls-id 100;
                    neighbor 192.0.2.1;
                }
                mesh-group pe2 {
                    vpls-id 200;
                    neighbor 192.0.2.2;
                }
            }
        }
    }
}
```

Verifying the H-VPLS Operation

Step-by-Step Procedure

This section describes the operational commands that you can use to validate that the H-VPLS is working as expected.

1. On Router PE1 and Router PE2, use the `show l2circuit connections` command to verify that the Layer 2 circuit to Router PE3 is `Up` and the Layer 2 circuit to Router PE4 is in `standby` mode.

   The output also shows the assigned label, virtual circuit ID, and the `ETHERNET` encapsulation type.

```
user@PE1> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.3
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.0(vc 100)        rmt   Up     Oct 18 15:55:07 2012           1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 299840, Outgoing label: 800001
      Negotiated PW status TLV: No
      Local interface: ge-2/0/5.0, Status: Up, Encapsulation: ETHERNET
Neighbor: 192.0.2.4
```

```
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.0(vc 100)        rmt   ST
```

```
user@PE2> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.3
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/6.0(vc 200)        rmt   Up     Oct 18 15:55:07 2012           1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 299872, Outgoing label: 800002
      Negotiated PW status TLV: No
      Local interface: ge-2/0/6.0, Status: Up, Encapsulation: ETHERNET
Neighbor: 192.0.2.4
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/6.0(vc 200)        rmt   ST
```

2. On Router PE1 and Router PE2, use the `show ldp neighbor` command to verify that the targeted LDP sessions have been created between the loopback interface to the primary and backup H-VPLS hub neighbors.

```
user@PE1> show ldp neighbor
Address            Interface          Label space ID         Hold time
10.10.3.2          ge-2/0/9.0           192.0.2.2:0               13
```

```
10.10.1.2          ge-2/0/10.0         192.0.2.3:0              10
192.0.2.3            lo0.0             192.0.2.3:0              36
192.0.2.4            lo0.0             192.0.2.4:0              39
10.10.9.2          ge-2/0/8.0          192.0.2.4:0              14
```

```
user@PE2> show ldp neighbor
Address            Interface          Label space ID         Hold time
10.10.3.1          ge-2/0/10.0         192.0.2.1:0              12
10.10.5.2          ge-2/0/9.0          192.0.2.4:0              11
10.10.4.1          ge-2/0/8.0          192.0.2.3:0              11
192.0.2.3            lo0.0             192.0.2.3:0              39
192.0.2.4            lo0.0             192.0.2.4:0              38
```

3. On Router PE3 and Router PE4, use the `show vpls connections` command to verify that the VPLS connection status is `Up` for both the LDP-based VPLS and the BGP-based VPLS Layer 2 circuits that are terminated.

```
user@PE3> show vpls connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch
```

```
Legend for interface status
Up -- operational
Dn -- down

Instance: H-VPLS
  BGP-VPLS State
  Local site: pe3 (3)
    connection-site           Type  St     Time last up          # Up trans
    4                         rmt   Up     Oct 18 15:58:39 2012           1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 800267, Outgoing label: 800266
      Local interface: vt-2/0/9.135266562, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls H-VPLS local site 3 remote site 4
  LDP-VPLS State
  Mesh-group connections: pe1
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.1(vpls-id 100)      rmt   Up     Oct 18 15:55:07 2012           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 299840
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.135266560, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.1 vpls-id 100
  Mesh-group connections: pe2
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.2(vpls-id 200)      rmt   Up     Oct 18 15:55:07 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800002, Outgoing label: 299872
      Negotiated PW status TLV: No
      Local interface: vt-2/0/8.135266561, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.2 vpls-id 200
user@PE4> show vpls connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
```

```
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: H-VPLS
  BGP-VPLS State
  Local site: pe4 (4)
    connection-site           Type  St     Time last up          # Up trans
    3                         rmt   Up     Oct 18 15:58:39 2012           1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800266, Outgoing label: 800267
      Local interface: vt-2/0/8.17826050, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls H-VPLS local site 4 remote site 3
  LDP-VPLS State
  Mesh-group connections: pe1
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.1(vpls-id 100)      rmt   Up     Oct 18 15:58:39 2012           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800002, Outgoing label: 299856
      Negotiated PW status TLV: No
      Local interface: vt-2/0/9.17826048, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.1 vpls-id 100
  Mesh-group connections: pe2
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.2(vpls-id 200)      rmt   Up     Oct 18 15:58:39 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800003, Outgoing label: 299888
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.17826049, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.2 vpls-id 200
```

4. On Router PE3 and Router PE4, use the `show vpls flood` command to verify that the H-VPLS PE router created a flood group for each spoke PE site.

```
user@PE3> show vpls flood
Name: H-VPLS
CEs: 1
VEs: 3
Flood Routes:
  Prefix    Type          Owner                NhType          NhIndex
  0x300cc/51 FLOOD_GRP_COMP_NH __ves__         comp            1376
  0x300cf/51 FLOOD_GRP_COMP_NH __all_ces__     comp            744
  0x300d5/51 FLOOD_GRP_COMP_NH pe1             comp            1702
  0x300d3/51 FLOOD_GRP_COMP_NH pe2             comp            1544
  0x30001/51 FLOOD_GRP_COMP_NH __re_flood__    comp            740
```

```
user@PE4> show vpls flood
Name: H-VPLS
CEs: 1
VEs: 3
Flood Routes:
  Prefix    Type          Owner                NhType          NhIndex
  0x300d1/51 FLOOD_GRP_COMP_NH __ves__         comp            1534
  0x300d0/51 FLOOD_GRP_COMP_NH __all_ces__     comp            753
  0x300d6/51 FLOOD_GRP_COMP_NH pe1             comp            1378
  0x300d4/51 FLOOD_GRP_COMP_NH pe2             comp            1695
  0x30002/51 FLOOD_GRP_COMP_NH __re_flood__    comp            750
```

5. On Router PE3 and Router PE4, use the `show vpls mac-table` command to verify that MAC addresses of the CE devices have been learned.

```
user@PE3> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : H-VPLS
 Bridging domain : __H-VPLS__, VLAN : NA
   MAC                 MAC      Logical          NH     RTR
   address             flags    interface        Index  ID
   00:21:59:0f:35:32   D        vt-2/0/8.135266560
   00:21:59:0f:35:33   D        ge-2/1/3.0
   00:21:59:0f:35:d4   D        vt-2/0/9.135266561
   00:21:59:0f:35:d5   D        vt-2/0/10.135266562
```

```
user@PE4> show vpls mac-table

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Logical system   : PE4
Routing instance : H-VPLS
 Bridging domain : __H-VPLS__, VLAN : NA
   MAC                 MAC      Logical          NH     RTR
   address             flags    interface        Index  ID
   00:21:59:0f:35:32   D        vt-2/0/8.17826050
   00:21:59:0f:35:33   D        vt-2/0/9.17826050
   00:21:59:0f:35:d4   D        vt-2/0/10.17826050
   00:21:59:0f:35:d5   D        ge-2/1/7.0
```

6. Make sure that the CE devices can ping each other.

```
user@CE1> ping 10.255.14.219 # ping sent from CE1 CE4
PING 10.255.14.219 (10.255.14.219): 56 data bytes
64 bytes from 10.255.14.219: icmp_seq=0 ttl=64 time=10.617 ms
64 bytes from 10.255.14.219: icmp_seq=1 ttl=64 time=9.224 ms
^C
--- 10.255.14.219 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 9.224/9.921/10.617/0.697 ms
```

```
user@CE2> ping 10.255.14.218 # ping sent from CE2 to CE3

PING 10.255.14.218 (10.255.14.218): 56 data bytes
64 bytes from 10.255.14.218: icmp_seq=0 ttl=64 time=1.151 ms
64 bytes from 10.255.14.218: icmp_seq=1 ttl=64 time=0.674 ms
^C
--- 10.255.14.218 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.674/0.913/1.151/0.238 ms
```

7. Check the relevant routing tables.

```
user@PE1> show route table l2circuit.0
l2circuit.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:NoCtrlWord:5:100:Local/96
                   *[L2CKT/7] 00:12:16, metric2 1
                    > to 10.10.1.2 via ge-2/0/10.0
192.0.2.3:NoCtrlWord:5:100:Remote/96
                   *[LDP/9] 00:12:16
                      Discard
192.0.2.4:NoCtrlWord:5:100:Local/96
                   *[L2CKT/7] 00:12:10, metric2 1
                    > to 10.10.9.2 via ge-2/0/8.0
192.0.2.4:NoCtrlWord:5:100:Remote/96
                   *[LDP/9] 00:12:15
                      Discard
```

```
user@PE2> show route table l2circuit.0
l2circuit.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:NoCtrlWord:5:200:Local/96
                   *[L2CKT/7] 00:13:13, metric2 1
                    > to 10.10.4.1 via ge-2/0/8.0
192.0.2.3:NoCtrlWord:5:200:Remote/96
                   *[LDP/9] 00:13:13
                      Discard
192.0.2.4:NoCtrlWord:5:200:Local/96
                   *[L2CKT/7] 00:13:13, metric2 1
                    > to 10.10.5.2 via ge-2/0/9.0
192.0.2.4:NoCtrlWord:5:200:Remote/96
                   *[LDP/9] 00:13:13
                      Discard
user@PE3> show route table H-VPLS.l2vpn.0

H-VPLS.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:33:3:1/96
```

```
                   *[L2VPN/170/-101] 03:19:26, metric2 1
                      Indirect
192.0.2.4:44:4:1/96
                   *[BGP/170] 03:15:45, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    > to 10.10.6.2 via ge-2/0/9.0
user@PE4> show route table H-VPLS.l2vpn.0

H-VPLS.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:33:3:1/96
                   *[BGP/170] 03:21:17, localpref 100, from 192.0.2.3
                      AS path: I, validation-state: unverified
                    > to 10.10.6.1 via ge-2/0/9.0
192.0.2.4:44:4:1/96
                   *[L2VPN/170/-101] 03:17:47, metric2 1
                      Indirect
```

Results

The configuration and verification parts of this example have been completed. The following section is for your reference.

The relevant sample configuration for Router PE1 follows.

Router PE1

```
interfaces {
    ge-2/0/5 {
        encapsulation ethernet-ccc;
        unit 0 {
            description to_CE1;
            family ccc;
        }
    }
    ge-2/0/8 {
        unit 0 {
            description to_PE4;
            family inet {
                address 10.10.9.1/30;
            }
            family mpls;
        }
    }
    ge-2/0/9 {
        unit 0 {
            description to_PE2;
            family inet {
                address 10.10.3.1/30;
            }
            family mpls;
        }
    }
    ge-2/0/10 {
        unit 0 {
            description to_PE3;
            family inet {
                address 10.10.1.1/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.1/24;
            }
        }
    }
}
protocols {
    mpls {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/8.0;
            interface ge-2/0/9.0;
            interface ge-2/0/10.0;
        }
    }
    ldp {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
        interface lo0.0;
    }
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-2/0/5.0 {
                virtual-circuit-id 100;
                backup-neighbor 192.0.2.4 {
                    standby;
                }
            }
        }
    }
}
```

The relevant sample configuration for Router PE2 follows.

Router PE2

```
interfaces {
    ge-2/0/6 {
        encapsulation ethernet-ccc;
        unit 0 {
            description to_CE2;
            family ccc;
        }
    }
    ge-2/0/8 {
        unit 0 {
            description to_PE3;
            family inet {
                address 10.10.4.2/30;
            }
            family mpls;
        }
    }
    ge-2/0/9 {
        unit 0 {
            description to_PE4;
            family inet {
                address 10.10.5.1/30;
            }
            family mpls;
        }
    }
    ge-2/0/10 {
        unit 0 {
            description to_PE1;
            family inet {
                address 10.10.3.2/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.2/24;
            }
        }
    }
}
protocols {
    mpls {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/8.0;
            interface ge-2/0/9.0;
            interface ge-2/0/10.0;
        }
    }
    ldp {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
        interface lo0.0;
    }
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-2/0/6.0 {
                virtual-circuit-id 200;
                backup-neighbor 192.0.2.4 {
                    standby;
                }
            }
        }
    }
}
```

The relevant sample configuration for Router PE3 follows.

Router PE3

```
interfaces {
    ge-2/0/8 {
        unit 0 {
            description to_PE2;
            family inet {
                address 10.10.4.1/30;
            }
            family mpls;
        }
    }
    ge-2/0/9 {
        unit 0 {
            description to_PE4;
            family inet {
                address 10.10.6.1/30;
            }
            family mpls;
        }
    }
    ge-2/0/10 {
        unit 0 {
            description to_PE1;
            family inet {
                address 10.10.1.2/30;
            }
            family mpls;
        }
    }
    ge-2/1/3 {
        encapsulation ethernet-vpls;
        unit 0 {
            description to_CE3;
            family vpls;
        }
    }
    lo0 {
        unit 0{
            family inet {
                address 192.0.2.3/24;
            }
        }
    }
}
protocols {
    mpls {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
    }
    bgp {
        group internal-peers {
            type internal;
            local-address 192.0.2.3;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.4;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/8.0;
            interface ge-2/0/9.0;
            interface ge-2/0/10.0;
        }
    }
    ldp {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
        interface lo0.0;
    }
}
routing-instances {
    H-VPLS {
        instance-type vpls;
        interface ge-2/1/3.0;
        route-distinguisher 192.0.2.3:33;
        vrf-target target:64510:2;
        protocols {
            vpls {
                site pe3 {
                    site-identifier 3;
                    interface ge-2/1/3.0;
                }
                mesh-group pe1 {
                    vpls-id 100;
                    neighbor 192.0.2.1;
                }
                mesh-group pe2 {
                    vpls-id 200;
                    neighbor 192.0.2.2;
                }
            }
        }
    }
}
routing-options {
    autonomous-system 64510;
}
```

The relevant sample configuration for Router PE4 follows.

Router PE4

```
interfaces {
    ge-2/0/8 {
        unit 0 {
            description to_PE3;
            family inet {
                address 10.10.6.2/30;
            }
            family mpls;
        }
    }
    ge-2/0/9 {
        unit 0 {
            description to_PE1;
            family inet {
                address 10.10.9.2/30;
            }
            family mpls;
        }
    }
    ge-2/0/10 {
        unit 0 {
            description to_PE2;
            family inet {
                address 10.10.5.2/30;
            }
            family mpls;
        }
    }
    ge-2/1/7 {
        encapsulation ethernet-vpls;
        unit 0 {
            description to_CE4;
            family vpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.4/24;
            }
        }
    }
}
protocols {
    mpls {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
    }
    bgp {
        group internal-peers {
            type internal;
            local-address 192.0.2.4;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.3;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/8.0;
            interface ge-2/0/9.0;
            interface ge-2/0/10.0;
        }
    }
    ldp {
        interface ge-2/0/8.0;
        interface ge-2/0/9.0;
        interface ge-2/0/10.0;
        interface lo0.0;
    }
}
routing-instances {
    H-VPLS {
        instance-type vpls;
        interface ge-2/1/7.0;
        route-distinguisher 192.0.2.4:44;
        vrf-target target:64510:2;
        protocols {
            vpls {
                site pe4 {
                    site-identifier 4;
                    interface ge-2/1/7.0;
                }
                mesh-group pe1 {
                    vpls-id 100;
                    neighbor 192.0.2.1;
                }
                mesh-group pe2 {
                    vpls-id 200;
                    neighbor 192.0.2.2;
                }
            }
        }
    }
}
routing-options {
    autonomous-system 64510;
}
```

The relevant sample configuration for Device CE1 follows.

Router CE1

```
interfaces {
    ge-2/0/8 {
        unit 0 {
            description to_PE1;
            family inet {
                address 172.16.0.1/24;
            }
        }
    }
    lo0 {
        unit 0{
            family inet {
                address 10.255.14.214/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/8.0;
        }
    }
}
```

The relevant sample configuration for Device CE2 follows.

Router CE2

```
interfaces {
    ge-2/1/5 {
        unit 0 {
            description to_PE2;
            family inet {
                address 172.16.0.2/24;
            }
        }
    }
    lo0 {
        unit 0{
            family inet {
                address 10.255.14.215/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/1/5.0;
        }
    }
}
```

The relevant sample configuration for Device CE3 follows.

Router CE3

```
interfaces {
    ge-2/0/9 {
        unit 0 {
            description to_PE3;
            family inet {
                address 172.16.0.3/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.255.14.218/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/0/9.0;
        }
    }
}
```

The relevant sample configuration for Device CE4 follows.

Router CE4

```
interfaces {
    ge-2/1/6 {
        unit 0 {
            description to_PE4;
            family inet {
                address 172.16.0.4/24;
            }
        }
    }
    lo0 {
        unit 0{
            family inet {
                address 10.255.14.219/32;
            }
        }
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-2/1/6.0;
        }
    }
}
```

RELATED DOCUMENTATION

Example: Configuring LDP-Based H-VPLS Using a Single Mesh Group to Terminate the Layer 2 Circuits | 833

Example: Configuring LDP-Based H-VPLS Using a Single Mesh Group to Terminate the Layer 2 Circuits

IN THIS SECTION

- Requirements | 834

- Overview and Topology | 834
- Configuration | 835

This example shows how to configure a single mesh group to terminate the Layer 2 circuits into an LDP-based VPLS. This is one type of hierarchical virtual private LAN service (H-VPLS) configuration possible in the Juniper Networks implementation. For information about the alternate type of configuration see "Example: Configuring BGP-Based H-VPLS Using Different Mesh Groups for Each Spoke Router" on page 805.

This example provides step-by-step configuration instructions and also provides steps for verifying and troubleshooting the configuration.

This example is organized into the following sections:

Requirements

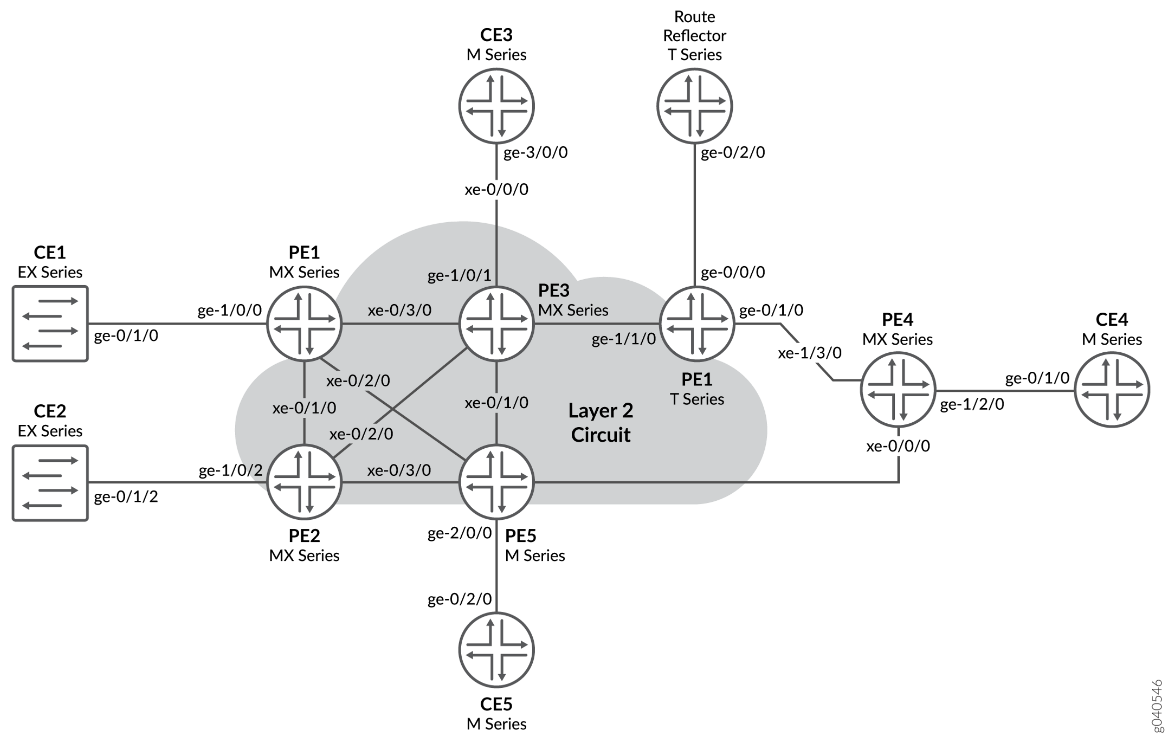
This example uses the following hardware components:

- Four MX Series 5G Universal Routing Platforms for Routers PE1, PE2, PE3, and PE4
- Two M Series Multiservice Edge Routers for Routers CE4 and PE5
- Two EX Series Ethernet Switches for Devices CE1 and CE2
- Two T Series Core Routers for Routers P1 and the route reflector

Overview and Topology

Figure 60 on page 835 shows the physical topology used in this example.

Figure 60: Physical Topology of H-VPLS using a Single Mesh Group



In Figure 60 on page 835:

- Local switching is used to switch traffic between Layer 2 circuit pseudowires from the different spoke PE routers.
- The spoke PE routers are configured with the same virtual circuit ID and VPLS ID pair in a mesh group.
- The spoke PE routers are configured in an LDP-signaled VPLS routing instance.
- The layer 2 circuits are terminated into the LDP-based VPLS.

Configuration

IN THIS SECTION

- Configuring the Spoke PE Routers | 836
- Configuring the Hub PE Router | 838
- Verification | 840

To configure a single mesh group to terminate the Layer 2 circuits into an LDP-based VPLS, perform the following tasks:

Configuring the Spoke PE Routers

Step-by-Step Procedure

Configure a single mesh group to terminate all the Layer 2 circuit pseudowires and enable local switching between the pseudowires.

1. On Router PE1, configure the Layer 2 circuit by including the `l2circuit` statement at the `[edit protocols]` hierarchy level. Include the `neighbor` statement and specify the IPv4 address of the hub PE router. Also configure the logical interface by including the `interface` statement and specify the interface connected to Router CE1.

   Configure the virtual circuit ID by including the `virtual-circuit-id` statement and specifying `100` as the ID value at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/0/0.0]` hierarchy level.

   Configure the backup neighbor by including the `backup-neighbor` statement and specifying the IPv4 address of the backup hub PE router. Router PE3 is the backup neighbor in this example. Also include the `standby` statement at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/0/0.0 backup-neighbor 192.0.2.3]` hierarchy level.

   ```
   [edit protocols]
   l2circuit {
       neighbor 192.0.2.5 {
           interface ge-1/0/0.0 {
               virtual-circuit-id 100;
               backup-neighbor 192.0.2.3 {
                   standby;
               }
           }
       }
   }
   ```

2. On Router PE2, configure the Layer 2 circuit by including the `l2circuit` statement at the `[edit protocols]` hierarchy level. Include the `neighbor` statement and specify the IPv4 address of the hub PE router. Configure the logical interface by including the `interface` statement and specifying the interface connected to Router CE2.

   Configure the virtual circuit ID by including the `virtual-circuit-id` statement and specifying `100` as the ID value at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/0/2.0]` hierarchy level. Include the `encapsulation` statement and specify `ethernet` as the type.

Configure the backup neighbor by including the `backup-neighbor` statement and specifying the IPv4 address of the backup hub PE router. Router PE3 is the backup neighbor in this example. Also include the `standby` statement at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/0/0.0 backup-neighbor 192.0.2.3]` hierarchy level.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.5 {
        interface ge-1/0/2.0 {
            virtual-circuit-id 100;
            encapsulation-type ethernet;
            backup-neighbor 192.0.2.3 {
                standby;
            }
        }
    }
}
```

3. On Router PE4, configure the Layer 2 circuit by including the `l2circuit` statement at the `[edit protocols]` hierarchy level. Include the `neighbor` statement and specify the IPv4 address of the hub PE router. Configure the logical interface by including the `interface` statement and specify the interface connected to Router CE4.

   Configure the virtual circuit ID by including the `virtual-circuit-id` statement and specifying `100` as the ID value at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/2/0.0]` hierarchy level.

   Configure the backup neighbor by including the `backup-neighbor` statement and specifying the IPv4 address of the backup hub PE router. Router PE3 is the backup neighbor in this example. Also include the `standby` statement at the `[edit protocols l2circuit neighbor 192.0.2.5 interface ge-1/2/0.0 backup-neighbor 192.0.2.3]` hierarchy level.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.5 {
        interface ge-1/2/0.0 {
            virtual-circuit-id 100;
            backup-neighbor 192.0.2.3 {
                standby;
            }
        }
    }
}
```

Configuring the Hub PE Router

Step-by-Step Procedure

Configure a single mesh group to terminate all the Layer 2 circuit pseudowires and enable local switching between the pseudowires.

1. On Router PE3, configure the Gigabit Ethernet interface connected to Router CE3 by including the `encapsulation` statement and specifying the `ethernet-vpls` option. Also configure the logical interface by including the `family` statement and specifying the `vpls` option.

```
[edit interfaces]
ge-1/0/1 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
```

2. On Router PE3, configure the logical loopback interface by including the `family` statement and specifying the `inet` option. Include the `address` statement and specify the IPv4 address for the interface.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
    }
}
```

3. On Router PE3, configure the LDP-based VPLS routing instance by including the `instance-type` statement at the `[edit routing-instances H-VPLS]` hierarchy level and specifying the `vpls` option. Include the `interface` statement and specify the Gigabit Ethernet interface connected to Router CE3.

Configure the VPLS protocol by including the `vpls` statement at the `[edit routing-instances H-VPLS protocols]` hierarchy level. Include the `no-tunnel-services` statement to enable the router to use an LSI interface.

```
[edit routing-instances]
H-VPLS {
    instance-type vpls;
    interface ge-1/0/1.0;
    protocols {
        vpls {
            no-tunnel-services;
        }
    }
}
```

4. On Router PE3, configure the mesh group by including the `mesh-group` statement at the `[edit routing-instances H-VPLS protocols vpls]` hierarchy level and specifying `L2-Circuits` as the name of the group. Include the `vpls-id` statement and specify `100` as the ID value. Include the `local-switching` statement to enable the router to switch traffic between the pseudowires.

   For each neighbor in the mesh group, include the `neighbor` statement and specify the IPv4 address of the spoke PE router.

```
[edit routing-instances H-VPLS protocols vpls]
mesh-group L2-Circuits {
    vpls-id 100;    <<< Same VPLS ID on all MTUs
    local-switching;    << Local-switching enabled
    neighbor 192.0.2.1;    << MTU IP addresses
    neighbor 192.0.2.2;
    neighbor 192.0.2.4;
}
```

Verification

Step-by-Step Procedure

1. On Router PE5, use the `show ldp neighbor` command to verify that LDP sessions have been created to each of the spoke PE routers.

```
user@PE5# show ldp neighbor
Address            Interface          Label space ID         Hold time
192.0.2.1            lo0.0              192.0.2.1:0                33
192.0.2.2            lo0.0              192.0.2.2:0                37
192.0.2.4            lo0.0              192.0.2.4:0                39
```

2. On Router PE5, use the `show vpls connections extensive` command to verify that the mesh group neighbor session is `Up`, that inbound and outbound labels have been assigned, that the VPLS ID is correct, and that the virtual tunnel interface is being used.

```
user@PE5# show vpls connections extensive
...
Instance: H-VPLS
    Number of local interfaces: 1
    Number of local interfaces up: 1
    Number of VE mesh-groups: 2
    Number of VE mesh-groups up: 1
    ge-2/0/0.0
    Mesh-group interfaces: L2-Circuits
      State: Up      ID: 2
      vt-2/1/0.1048848    Intf - vpls H-VPLS neighbor 192.0.2.4 vpls-id 100
      vt-2/1/0.1048849    Intf - vpls H-VPLS neighbor 192.0.2.2 vpls-id 100
      vt-2/1/0.1048850    Intf - vpls H-VPLS neighbor 192.0.2.1 vpls-id 100
    Mesh-group interfaces: __ves__
      State: Dn      ID: 0
  Mesh-group connections: L2-Circuits
    Neighbor                  Type  St     Time last up          # Up trans
    192.0.2.4(vpls-id 100)      rmt   Up     Jan  3 16:46:26 2010           1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 800011, Outgoing label: 301088
      Local interface: vt-2/1/0.1048848, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.4 vpls-id 100
    Connection History:
        Jan  3 16:46:26 2010  status update timer
        Jan  3 16:46:26 2010  PE route changed
        Jan  3 16:46:26 2010  In lbl Update                    800011
        Jan  3 16:46:26 2010  Out lbl Update                   301088
        Jan  3 16:46:26 2010  In lbl Update                    800011
        Jan  3 16:46:26 2010  loc intf up            vt-2/1/0.1048848
    192.0.2.2(vpls-id 100)      rmt   Up     Jan  3 16:46:26 2010           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800010, Outgoing label: 301488
      Local interface: vt-2/1/0.1048849, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.2 vpls-id 100
    Connection History:
        Jan  3 16:46:26 2010  status update timer
        Jan  3 16:46:26 2010  PE route changed
        Jan  3 16:46:26 2010  In lbl Update                    800010
        Jan  3 16:46:26 2010  Out lbl Update                   301488
        Jan  3 16:46:26 2010  In lbl Update                    800010
        Jan  3 16:46:26 2010  loc intf up            vt-2/1/0.1048849
    192.0.2.1(vpls-id 100)      rmt   Up     Jan  3 16:46:26 2010           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800009, Outgoing label: 301296
      Local interface: vt-2/1/0.1048850, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls H-VPLS neighbor 192.0.2.1 vpls-id 100
    Connection History:
        Jan  3 16:46:26 2010  status update timer
        Jan  3 16:46:26 2010  PE route changed
        Jan  3 16:46:26 2010  In lbl Update                    800009
        Jan  3 16:46:26 2010  Out lbl Update                   301296
        Jan  3 16:46:26 2010  In lbl Update                    800009
        Jan  3 16:46:26 2010  loc intf up            vt-2/1/0.1048850
```

RELATED DOCUMENTATION

Example: Configuring BGP-Based H-VPLS Using Different Mesh Groups for Each Spoke Router | 805

Example: Configuring H-VPLS With VLANs

IN THIS SECTION

- Requirements | 842
- Overview | 842
- Configuration | 844
- Verification | 855

This example shows how to configure the hierarchical virtual private LAN service (H-VPLS). VLANs are configured in this example.

Requirements

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

Overview

H-VPLS uses LDP-based VPLS to signal and establish pseudowires. LDP-based VPLS is defined in RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*. RFC 4762 also defines a hierarchical mode of operation for LDP VPLS called H-VPLS.

VPLS and H-VPLS are different with respect to scaling. VPLS requires a full mesh of tunnel label-switched paths (LSPs) among all of the provider edge (PE) routers that participate in the VPLS service. For each VPLS service, n*(n-1)/2 pseudowires must be set up between the PE routers. In contrast, H-VPLS partitions the network into several edge domains that are interconnected using an MPLS core. Each edge device only needs to learn of one local PE device and therefore needs less routing table support. This has the potential to allow service providers to use relatively less costly devices (such as EX Series switches) at the customer edge.

NOTE: As alternatives to H-VPLS, Juniper Networks offers other ways to address VPLS scalability. For more information, see Application Note: Demystifying H-VPLS.

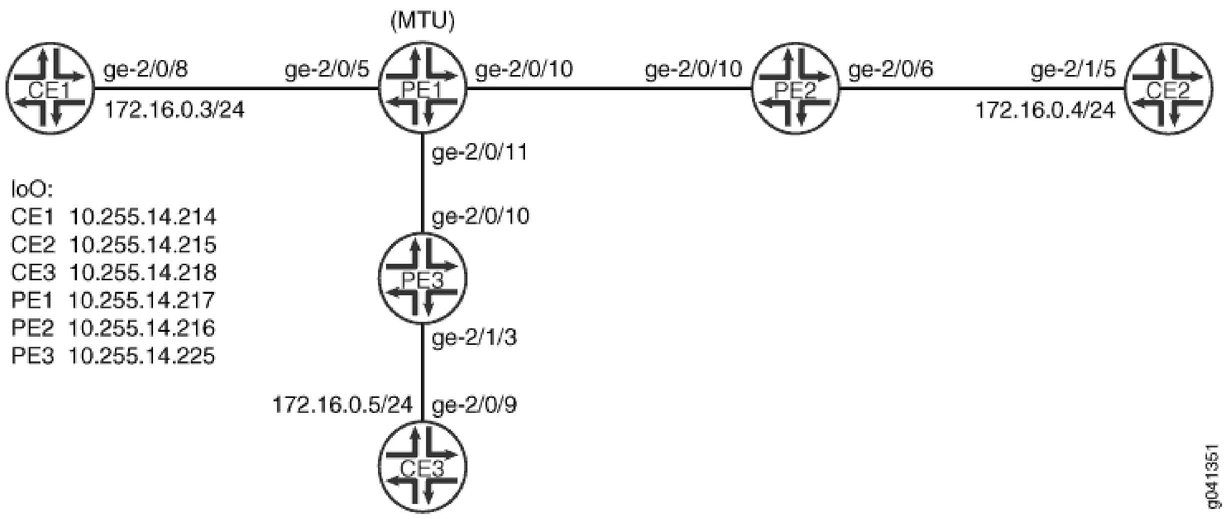
H-VPLS defines two roles or functionalities:

- PE-r—PE device that runs VPLS with other PE-r devices, but which also has pseudowires (it can be based on QinQ access) with another device called a multi-tenant unit (MTU), which provides the access layer.

- MTU—PE device that represents the access layer on the H-VPLS architecture and establishes pseudowires to one or more PE-r devices through which VPLS traffic is forwarded.

Figure 61 on page 843 shows the topology used in this example.

Figure 61: Basic H-VPLS With One MTU and Two PE-r Devices



The example shows one MTU (Device PE1) connected to two PE-r devices (Device PE2 and Device PE3).

The pseudowire between Device PE1 and Device PE3 is the primary or working path. The pseudowire between Device PE1 and Device PE2 is the backup path.

To support VLANs with H-VPLS, you must include the `output-vlan-map swap` statement in the configuration of the MTU device as a workaround to prevent a VLAN ID mismatch. Otherwise, the PE-r devices report a VLAN ID mismatch, as shown here:

```
user@PE2> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
```

```
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: customer
  VPLS-id: 601
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.14.217(vpls-id 601) rmt  VM
```

"CLI Quick Configuration" on page 844 shows the configuration for all of the devices in Figure 61 on page 843. The section "Step-by-Step Procedure" on page 847 describes the steps on Device PE1 and Device PE2.

Configuration

IN THIS SECTION

- Procedure | 844

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 PE1

```
set interfaces ge-2/0/5 vlan-tagging
set interfaces ge-2/0/5 encapsulation vlan-ccc
set interfaces ge-2/0/5 unit 601 encapsulation vlan-ccc
set interfaces ge-2/0/5 unit 601 vlan-id 601
set interfaces ge-2/0/5 unit 601 output-vlan-map swap
set interfaces ge-2/0/5 unit 601 family ccc
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/0/11 unit 0 family inet address 192.0.2.3/24
set interfaces ge-2/0/11 unit 0 family iso
set interfaces ge-2/0/11 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.217/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4217.00
set protocols mpls interface ge-2/0/10.0
set protocols mpls interface ge-2/0/11.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface ge-2/0/11.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface ge-2/0/11.0
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.601 virtual-circuit-id 601
set protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.601 encapsulation-type 
ethernet-vlan
set protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.601 backup-neighbor 
10.255.14.216 standby
set routing-options router-id 10.255.14.217
```

Device PE2

```
set interfaces ge-2/0/6 vlan-tagging
set interfaces ge-2/0/6 encapsulation vlan-vpls
set interfaces ge-2/0/6 unit 601 encapsulation vlan-vpls
set interfaces ge-2/0/6 unit 601 vlan-id 601
set interfaces ge-2/0/6 unit 601 family vpls
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.4/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.216/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4216.00
set protocols mpls interface ge-2/0/10.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances customer instance-type vpls
set routing-instances customer interface ge-2/0/6.601
set routing-instances customer protocols vpls vpls-id 601
set routing-instances customer protocols vpls neighbor 10.255.14.217 encapsulation-type ethernet-
vlan
set routing-options router-id 10.255.14.216
```

Device PE3

```
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.5/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/1/3 vlan-tagging
set interfaces ge-2/1/3 encapsulation vlan-vpls
set interfaces ge-2/1/3 unit 601 encapsulation vlan-vpls
set interfaces ge-2/1/3 unit 601 vlan-id 601
set interfaces ge-2/1/3 unit 601 family vpls
set interfaces lo0 unit 0 family inet address 10.255.14.225/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4225.00
set protocols mpls interface ge-2/0/10.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances customer instance-type vpls
set routing-instances customer interface ge-2/1/3.601
set routing-instances customer protocols vpls vpls-id 601
set routing-instances customer protocols vpls neighbor 10.255.14.217 encapsulation-type ethernet-
vlan
set routing-options router-id 10.255.14.225
```

Device CE1

```
set interfaces ge-2/0/8 vlan-tagging
set interfaces ge-2/0/8 unit 601 vlan-id 601
set interfaces ge-2/0/8 unit 601 family inet address 172.16.0.3/24
set interfaces lo0 unit 0 family inet address 10.255.14.214/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/8.601
```

Device CE2

```
set interfaces ge-2/1/5 vlan-tagging
set interfaces ge-2/1/5 unit 601 vlan-id 601
set interfaces ge-2/1/5 unit 601 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 10.255.14.215/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/5.601
```

Device CE3

```
set interfaces ge-2/0/9 vlan-tagging
set interfaces ge-2/0/9 unit 601 vlan-id 601
set interfaces ge-2/0/9 unit 601 family inet address 172.16.0.5/24
set interfaces lo0 unit 0 family inet address 10.255.14.218/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/9.601
```

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 H-VPLS on the MTU device:

1. Configure the interfaces.

   On the MTU device interface that connects to the customer edge, configure one of the circuit cross-connect (CCC) encapsulation types and the CCC address family. This enables Layer 2 circuits.

On the core-facing interfaces, enable MPLS labels. The ISO address is needed as well on the core-facing interfaces because IS-IS is used in the core.

```
[edit interfaces]
user@PE1# set ge-2/0/5 vlan-tagging
user@PE1# set ge-2/0/5 encapsulation vlan-ccc
user@PE1# set ge-2/0/5 unit 601 family ccc
user@PE1# set ge-2/0/5 unit 601 encapsulation vlan-ccc
user@PE1# set ge-2/0/5 unit 601 vlan-id 601
user@PE1# set ge-2/0/5 unit 601 output-vlan-map swap
user@PE1# set ge-2/0/10 unit 0 family inet address 192.0.2.2/24
user@PE1# set ge-2/0/10 unit 0 family iso
user@PE1# set ge-2/0/10 unit 0 family mpls
user@PE1# set ge-2/0/11 unit 0 family inet address 192.0.2.3/24
user@PE1# set ge-2/0/11 unit 0 family iso
user@PE1# set ge-2/0/11 unit 0 family mpls
user@PE1# set lo0 unit 0 family inet address 10.255.14.217/32
user@PE1# set lo0 unit 0 family iso address 49.0001.0102.5501.4217.00
```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

```
[edit protocols mpls]
user@PE1# set interface ge-2/0/10.0
user@PE1# set interface ge-2/0/11.0
[edit protocols ldp ]
user@PE1# set interface ge-2/0/10.0
user@PE1# set interface ge-2/0/11.0
user@PE1# set interface lo0.0
```

3. Enable routing on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

```
[edit protocols isis]
user@PE1# set level 1 disable
user@PE1# set interface ge-2/0/10.0
```

```
user@PE1# set interface ge-2/0/11.0
user@PE1# set interface lo0.0
```

4. Configure the Layer 2 circuit.

   The neighbor 10.255.14.225 is Device PE3's loopback interface address. This sets up the working path.

   The neighbor 10.255.14.216 is Device PE2's loopback interface address. This sets up the backup path.

   The virtual circuit ID must match the VPLS ID that is configured on Device PE2 and Device PE3.

```
[edit protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.601]
user@PE1# set virtual-circuit-id 601
user@PE1# set encapsulation-type ethernet-vlan
user@PE1# set backup-neighbor 10.255.14.216 standby
```

5. Configure the router ID.

```
[edit routing-options]
user@PE1# set router-id 10.255.14.217
```

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 H-VPLS on the MTU device:

1. Configure the interfaces.

   On the PE-r device interface that connects to the customer edge, configure one of the VPLS encapsulation types and the VPLS address family. This enables VPLS.

   On the core-facing interfaces, enable MPLS labels. The ISO address is needed as well on the core-facing interfaces because IS-IS is used in the core.

```
[edit interfaces]
user@PE2# set ge-2/0/6 vlan-tagging
user@PE2# set ge-2/0/6 encapsulation vlan-vpls
```

```
user@PE2# set ge-2/0/6 unit 601 encapsulation vlan-vpls
user@PE2# set ge-2/0/6 unit 601 vlan-id 601
user@PE2# set ge-2/0/6 unit 601 family vpls
user@PE2# set ge-2/0/10 unit 0 family inet address 192.0.2.4/24
user@PE2# set ge-2/0/10 unit 0 family iso
user@PE2# set ge-2/0/10 unit 0 family mpls
user@PE2# set lo0 unit 0 family inet address 10.255.14.216/32
user@PE2# set lo0 unit 0 family iso address 49.0001.0102.5501.4216.00
```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

```
[edit protocols mpls]
user@PE2# set interface ge-2/0/10.0
[edit protocols ldp ]
user@PE2# set interface ge-2/0/10.0
user@PE2# set interface lo0.0
```

3. Enable routing on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

```
[edit protocols isis]
user@PE2# set level 1 disable
user@PE2# set interface ge-2/0/10.0
user@PE2# set interface lo0.0
```

4. Configure VPLS.

   The `neighbor 10.255.14.217` statement points to Device PE1’s loopback interface address.

   The VPLS ID must match the virtual circuit ID that is configured on the MTU (Device PE1).

```
[edit routing-instances customer]
user@PE2# set instance-type vpls
user@PE2# set interface ge-2/0/6.601
user@PE2# set protocols vpls vpls-id 601
user@PE2# set protocols vpls neighbor 10.255.14.217 encapsulation-type ethernet-vlan
```

5. Configure the router ID.

```
[edit routing-options]
user@PE2# set router-id 10.255.14.216
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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.

Device PE1

```
user@PE1# show interfaces
ge-2/0/5 {
    vlan-tagging;
    encapsulation vlan-ccc;
    unit 601 {
        encapsulation vlan-ccc;
        vlan-id 601;
        output-vlan-map swap;
        family ccc;
    }
}
ge-2/0/10 {
    unit 0 {
        family inet {
            address 192.0.2.2/24;
        }
        family iso;
        family mpls;
    }
}
ge-2/0/11 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.14.217/32;
        }
        family iso {
            address 49.0001.0102.5501.4217.00;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
}
isis {
    level 1 disable;
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
    interface lo0.0;
}
ldp {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
    interface lo0.0;
}
l2circuit {
    neighbor 10.255.14.225 {
        interface ge-2/0/5.601 {
            virtual-circuit-id 601;
            encapsulation-type ethernet-vlan;
            backup-neighbor 10.255.14.216 {
                standby;
            }
        }
    }
}
```

```
user@PE1# show routing-options
router-id 10.255.14.217;
```

Device PE2

```
user@PE2# show interfaces
ge-2/0/6 {
    vlan-tagging;
    encapsulation vlan-vpls;
    unit 601 {
        encapsulation vlan-vpls;
        vlan-id 601;
        family vpls;
    }
}
ge-2/0/10 {
    unit 0 {
        family inet {
            address 192.0.2.4/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.14.216/32;
        }
        family iso {
            address 49.0001.0102.5501.4216.00;
        }
    }
}
```

```
user@PE2# show protocols
mpls {
    interface ge-2/0/10.0;
}
isis {
    level 1 disable;
    interface ge-2/0/10.0;
    interface lo0.0;
}
ldp {
    interface ge-2/0/10.0;
    interface lo0.0;
}
```

```
user@PE2# show routing-instances
customer {
    instance-type vpls;
    interface ge-2/0/6.601;
    protocols {
        vpls {
            vpls-id 601;
            neighbor 10.255.14.217 {
                encapsulation-type ethernet-vlan;
            }
        }
    }
}
```

```
user@PE2# show routing-options
router-id 10.255.14.216;
```

If you are done configuring the devices, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying the Layer 2 Circuit | 855
- Checking the VPLS Connections | 856
- Checking Connectivity | 858
- Manually Triggering a Switch from the Active Pseudowire to the Redundant Pseudowire | 859

Confirm that the configuration is working properly.

Verifying the Layer 2 Circuit

Purpose

Verify that Layer 2 circuit is operational on the MTU device.

Action

From operational mode, enter the `show l2circuit connections` command.

```
user@PE1> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
```

```
Up -- operational
Dn -- down
Neighbor: 10.255.14.216
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.601(vc 601)      rmt   Up     Oct  9 16:28:58 2012           1
      Remote PE: 10.255.14.216, Negotiated control-word: No
      Incoming label: 299904, Outgoing label: 800001
      Negotiated PW status TLV: No
      Local interface: ge-2/0/5.601, Status: Up, Encapsulation: VLAN
Neighbor: 10.255.14.225
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.601(vc 601)      rmt   ST
```

Meaning

As expected, the Layer 2 circuit connection to Device PE3 is operational, and the connection to Device PE2 is in standby mode.

Checking the VPLS Connections

Purpose

Verify that the VPLS connections are operational on the PE-r devices.

Action

From operational mode, enter the `show vpls connections` command.

```
user@PE2> show vpls connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
```

```
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: customer
  VPLS-id: 601
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.14.217(vpls-id 601) rmt  Up     Oct  9 16:29:02 2012           1
      Remote PE: 10.255.14.217, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 299904
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.84934914, Status: Up, Encapsulation: VLAN
        Description: Intf - vpls customer neighbor 10.255.14.217 vpls-id 601
```

```
user@PE3> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
```

```
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: customer
  VPLS-id: 601
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.14.217(vpls-id 601) rmt  Up     Oct  9 16:29:02 2012           1
      Remote PE: 10.255.14.217, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 299920
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.68157698, Status: Up, Encapsulation: VLAN
        Description: Intf - vpls customer neighbor 10.255.14.217 vpls-id 601
```

Meaning

As expected, the VPLS connections are operational on both PE-r devices.

Checking Connectivity

Purpose

Verify that Device CE1 can ping Device CE3.

Action

```
user@CE1> **ping 10.255.14.218**
PING 10.255.14.218 (10.255.14.218): 56 data bytes
64 bytes from 10.255.14.218: icmp_seq=0 ttl=64 time=0.858 ms
64 bytes from 10.255.14.218: icmp_seq=1 ttl=64 time=0.527 ms
64 bytes from 10.255.14.218: icmp_seq=2 ttl=64 time=0.670 ms
^C
--- 10.255.14.218 ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.527/0.685/0.858/0.136 ms
```

Meaning

The output shows that H-VPLS is operational.

Manually Triggering a Switch from the Active Pseudowire to the Redundant Pseudowire

Purpose

Make sure that the pseudowire between Device PE1 and Device PE2 becomes operational.

Action

```
user@CE1> request l2circuit-switchover virtual-circuit-id 601 neighbor 10.255.14.225
user@CE1> ping 10.255.14.215
PING 10.255.14.215 (10.255.14.215): 56 data bytes
64 bytes from 10.255.14.215: icmp_seq=0 ttl=64 time=0.738 ms
64 bytes from 10.255.14.215: icmp_seq=1 ttl=64 time=0.627 ms
64 bytes from 10.255.14.215: icmp_seq=2 ttl=64 time=0.629 ms
^C
--- 10.255.14.215 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.627/0.665/0.738/0.052 ms
```

Meaning

The successful ping from Device CE1 to Device CE2 shows that the pseudowire between Device PE1 and PE2 is operational. Now, if you ping Device CE3 from Device CE1, the ping should fail.

RELATED DOCUMENTATION

Application Note: Demystifying H-VPLS

Example: Configuring H-VPLS Without VLANs | **860**

Redundant Pseudowires for Layer 2 Circuits and VPLS | **224**

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | **226**

Example: Configuring H-VPLS Without VLANs

IN THIS SECTION

- Requirements | 860
- Overview | 860
- Configuration | 861
- Verification | 871

This example shows how to configure the hierarchical virtual private LAN service (H-VPLS). No VLANs are configured in this example.

Requirements

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

Overview

H-VPLS uses LDP-based VPLS to signal and establish pseudowires. LDP-based VPLS is defined in RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*. RFC 4762 also defines a hierarchical mode of operation for LDP VPLS called H-VPLS.

VPLS and H-VPLS are different with respect to scaling. VPLS requires a full mesh of tunnel label-switched paths (LSPs) among all of the provider edge (PE) routers that participate in the VPLS service. For each VPLS service, n*(n-1)/2 pseudowires must be set up between the PE routers. In contrast, H-VPLS partitions the network into several edge domains that are interconnected using an MPLS core. Each edge device only needs to learn of one local PE device and therefore needs less routing table support. This has the potential to allow service providers to use relatively less costly devices (such as EX Series switches) at the customer edge.

NOTE: As alternatives to H-VPLS, Juniper Networks offers other ways to address VPLS scalability. For more information, see Application Note: Demystifying H-VPLS.

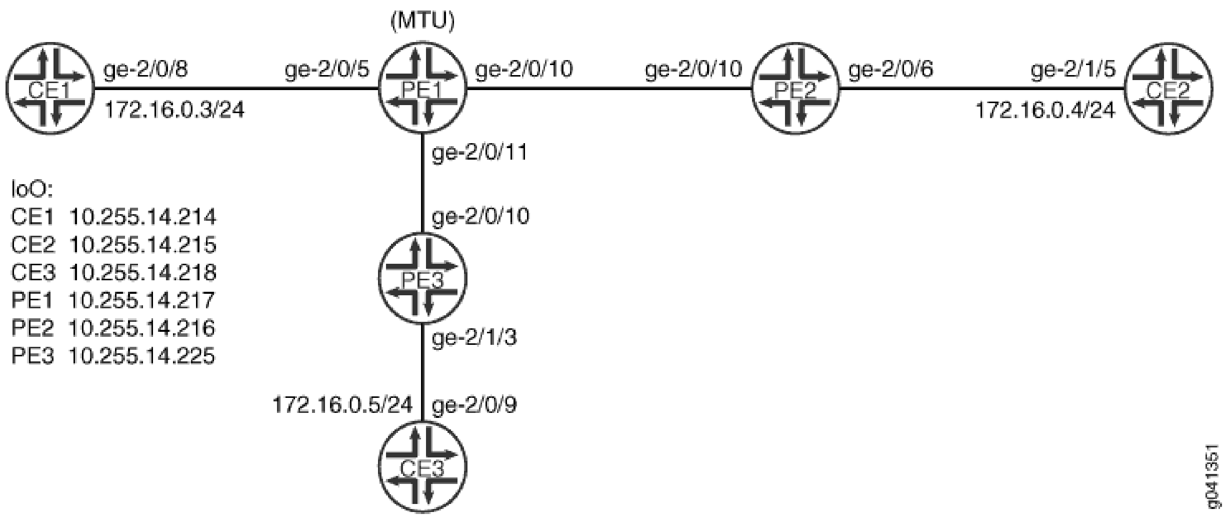
H-VPLS defines two roles or functionalities:

- PE-r—PE device that runs VPLS with other PE-r devices, but which also has pseudowires (it can be based on QinQ access) with another device called a multi-tenant unit (MTU), which provides the access layer.

- MTU—PE device that represents the access layer on the H-VPLS architecture and establishes pseudowires to one or more PE-r devices through which VPLS traffic is forwarded.

Figure 62 on page 861 shows the topology used in this example.

Figure 62: Basic H-VPLS With One MTU and Two PE-r Devices



The example shows one MTU (Device PE1) connected to two PE-r devices (Device PE2 and Device PE3).

The pseudowire between Device PE1 and Device PE3 is the primary or working path. The pseudowire between Device PE1 and Device PE2 is the backup path.

"CLI Quick Configuration" on page 862 shows the configuration for all of the devices in Figure 62 on page 861. The section "Step-by-Step Procedure" on page 864 describes the steps on Device PE1 and Device PE2.

Configuration

IN THIS SECTION

- Procedure | 862

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 PE1

```
set interfaces ge-2/0/5 encapsulation ethernet-ccc
set interfaces ge-2/0/5 unit 0 family ccc
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.2/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/0/11 unit 0 family inet address 192.0.2.3/24
set interfaces ge-2/0/11 unit 0 family iso
set interfaces ge-2/0/11 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.217/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4217.00
set protocols mpls interface ge-2/0/10.0
set protocols mpls interface ge-2/0/11.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface ge-2/0/11.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface ge-2/0/11.0
set protocols ldp interface lo0.0
set protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.0 virtual-circuit-id 601
set protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.0 backup-neighbor
10.255.14.216 standby
set routing-options router-id 10.255.14.217
```

Device PE2

```
set interfaces ge-2/0/6 encapsulation ethernet-vpls
set interfaces ge-2/0/6 unit 0 family vpls
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.4/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.216/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4216.00
set protocols mpls interface ge-2/0/10.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances customer instance-type vpls
set routing-instances customer interface ge-2/0/6.0
set routing-instances customer protocols vpls vpls-id 601
set routing-instances customer protocols vpls neighbor 10.255.14.217
set routing-options router-id 10.255.14.216
```

Device PE3

```
set interfaces ge-2/1/3 encapsulation ethernet-vpls
set interfaces ge-2/1/3 unit 0 family vpls
set interfaces ge-2/0/10 unit 0 family inet address 192.0.2.5/24
set interfaces ge-2/0/10 unit 0 family iso
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.14.225/32
set interfaces lo0 unit 0 family iso address 49.0001.0102.5501.4225.00
set protocols mpls interface ge-2/0/10.0
set protocols isis level 1 disable
set protocols isis interface ge-2/0/10.0
set protocols isis interface lo0.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances customer instance-type vpls
set routing-instances customer interface ge-2/1/3.0
set routing-instances customer protocols vpls vpls-id 601
set routing-instances customer protocols vpls neighbor 10.255.14.217
set routing-options router-id 10.255.14.225
```

Device CE1

```
set interfaces ge-2/0/8 unit 0 family inet address 172.16.0.3/24
set interfaces lo0 unit 0 family inet address 10.255.14.214/32
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 CE2

```
set interfaces ge-2/1/5 unit 0 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 10.255.14.215/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/5.0
```

Device CE3

```
set interfaces ge-2/0/9 unit 0 family inet address 172.16.0.5/24
set interfaces lo0 unit 0 family inet address 10.255.14.218/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/9.0
```

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 CLI User Guide.

To configure H-VPLS on the MTU device:

1. Configure the interfaces.

   On the MTU device interface that connects to the customer edge, configure one of the circuit cross-connect (CCC) encapsulation types and the CCC address family. This enables Layer 2 circuits.

   On the core-facing interfaces, enable MPLS labels. The ISO address is needed as well on the core-facing interfaces because IS-IS is used in the core.

   ```
   [edit interfaces]
   user@PE1# set ge-2/0/5 encapsulation ethernet-ccc
   user@PE1# set ge-2/0/5 unit 0 family ccc
   user@PE1# set ge-2/0/10 unit 0 family inet address 192.0.2.2/24
   user@PE1# set ge-2/0/10 unit 0 family iso
   user@PE1# set ge-2/0/10 unit 0 family mpls
   user@PE1# set ge-2/0/11 unit 0 family inet address 192.0.2.3/24
   user@PE1# set ge-2/0/11 unit 0 family iso
   user@PE1# set ge-2/0/11 unit 0 family mpls
   user@PE1# set lo0 unit 0 family inet address 10.255.14.217/32
   user@PE1# set lo0 unit 0 family iso address 49.0001.0102.5501.4217.00
   ```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

   ```
   [edit protocols mpls]
   user@PE1# set interface ge-2/0/10.0
   user@PE1# set interface ge-2/0/11.0
   [edit protocols ldp ]
   user@PE1# set interface ge-2/0/10.0
   user@PE1# set interface ge-2/0/11.0
   user@PE1# set interface lo0.0
   ```

3. Enable routing on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

   ```
   [edit protocols isis]
   user@PE1# set level 1 disable
   user@PE1# set interface ge-2/0/10.0
   user@PE1# set interface ge-2/0/11.0
   user@PE1# set interface lo0.0
   ```

4. Configure the Layer 2 circuit.

   The neighbor 10.255.14.225 is Device PE3’s loopback interface address. This sets up the working path.

   The neighbor 10.255.14.216 is Device PE2’s loopback interface address. This sets up the backup path.

   The virtual circuit ID must match the VPLS ID that is configured on Device PE2 and Device PE3.

   ```
   [edit protocols l2circuit neighbor 10.255.14.225 interface ge-2/0/5.0]
   user@PE1# set virtual-circuit-id 601
   user@PE1# set backup-neighbor 10.255.14.216 standby
   ```

5. Configure the router ID.

```
[edit routing-options]
user@PE1# set router-id 10.255.14.217
```

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 H-VPLS on the MTU device:

1. Configure the interfaces.

   On the PE-r device interface that connects to the customer edge, configure one of the VPLS encapsulation types and the VPLS address family. This enables VPLS.

   On the core-facing interfaces, enable MPLS labels. The ISO address is needed as well on the core-facing interfaces because IS-IS is used in the core.

```
[edit interfaces]
user@PE2# set ge-2/0/6 encapsulation ethernet-vpls
user@PE2# set ge-2/0/6 unit 0 family vpls
user@PE2# set ge-2/0/10 unit 0 family inet address 192.0.2.4/24
user@PE2# set ge-2/0/10 unit 0 family iso
user@PE2# set ge-2/0/10 unit 0 family mpls
user@PE2# set lo0 unit 0 family inet address 10.255.14.216/32
user@PE2# set lo0 unit 0 family iso address 49.0001.0102.5501.4216.00
```

2. Enable MPLS and LDP on the interfaces.

   On the MTU device interfaces that connect to other PE devices, configure MPLS and LDP.

```
[edit protocols mpls]
user@PE2# set interface ge-2/0/10.0
[edit protocols ldp ]
user@PE2# set interface ge-2/0/10.0
user@PE2# set interface lo0.0
```

3. Enable routing on the interfaces.

On the MTU device interfaces that connect to other PE devices, configure an interior gateway protocol (IGP), such as OSPF or IS-IS.

```
[edit protocols isis]
user@PE2# set level 1 disable
user@PE2# set interface ge-2/0/10.0
user@PE2# set interface lo0.0
```

4. Configure VPLS.

   The `neighbor 10.255.14.217` statement points to Device PE1's loopback interface address.

   The VPLS ID must match the virtual circuit ID that is configured on the MTU (Device PE1).

```
[edit routing-instances customer]
user@PE2# set instance-type vpls
user@PE2# set interface ge-2/0/6.0
user@PE2# set protocols vpls vpls-id 601
user@PE2# set protocols vpls neighbor 10.255.14.217
```

5. Configure the router ID.

```
[edit routing-options]
user@PE2# set router-id 10.255.14.216
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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.

Device PE1

```
user@PE1# show interfaces
ge-2/0/5 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
```

```
ge-2/0/10 {
    unit 0 {
        family inet {
            address 192.0.2.2/24;
        }
        family iso;
        family mpls;
    }
}
ge-2/0/11 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.14.217/32;
        }
        family iso {
            address 49.0001.0102.5501.4217.00;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
}
isis {
    level 1 disable;
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
    interface lo0.0;
}
ldp {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
    interface lo0.0;
}
l2circuit {
    neighbor 10.255.14.225 {
        interface ge-2/0/5.0 {
            virtual-circuit-id 601;
            backup-neighbor 10.255.14.216 {
                standby;
            }
        }
    }
}
```

```
user@PE1# show routing-options
router-id 10.255.14.217;
```

Device PE2

```
user@PE2# show interfaces
ge-2/0/6 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
ge-2/0/10 {
    unit 0 {
        family inet {
            address 192.0.2.4/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.255.14.216/32;
        }
        family iso {
            address 49.0001.0102.5501.4216.00;
        }
    }
}
```

```
user@PE2# show protocols
mpls {
    interface ge-2/0/10.0;
}
isis {
    level 1 disable;
    interface ge-2/0/10.0;
    interface lo0.0;
}
ldp {
    interface ge-2/0/10.0;
    interface lo0.0;
}
```

```
user@PE2# show routing-instances
customer {
    instance-type vpls;
    interface ge-2/0/6.0;
    protocols {
        vpls {
            vpls-id 601;
            neighbor 10.255.14.217;
        }
    }
}
```

```
user@PE2# show routing-options
router-id 10.255.14.216;
```

If you are done configuring the devices, enter **commit** from configuration mode.

Verification

IN THIS SECTION

- Verifying the Layer 2 Circuit | 871
- Checking the VPLS Connections | 872
- Checking Connectivity | 874
- Manually Triggering a Switch from the Active Pseudowire to the Redundant Pseudowire | 875

Confirm that the configuration is working properly.

Verifying the Layer 2 Circuit

Purpose

Verify that the Layer 2 circuit is operational on the MTU device.

Action

From operational mode, enter the `show l2circuit connections` command.

```
user@PE1> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
```

```
Up -- operational
Dn -- down
Neighbor: 10.255.14.216
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.0(vc 601)        rmt   ST
Neighbor: 10.255.14.225
    Interface                 Type  St     Time last up          # Up trans
    ge-2/0/5.0(vc 601)        rmt   Up     Oct  5 19:38:15 2012           1
      Remote PE: 10.255.14.225, Negotiated control-word: No
      Incoming label: 299872, Outgoing label: 800001
      Negotiated PW status TLV: No
      Local interface: ge-2/0/5.0, Status: Up, Encapsulation: ETHERNET
```

Meaning

As expected, the Layer 2 circuit connection to Device PE3 is operational, and the connection to Device PE2 is in standby mode.

Checking the VPLS Connections

Purpose

Verify that the VPLS connections are operational on the PE-r devices.

Action

From operational mode, enter the `show vpls connections` command.

```
user@PE2> show vpls connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
```

```
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: customer
  VPLS-id: 601
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.14.217(vpls-id 601) rmt  Up     Oct  8 14:46:54 2012           1
      Remote PE: 10.255.14.217, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 299856
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.84934913, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls customer neighbor 10.255.14.217 vpls-id 601
```

```
user@PE3> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
```

```
MM -- MTU mismatch              MI -- Mesh-Group ID not available
BK -- Backup connection         ST -- Standby connection
PF -- Profile parse failure     PB -- Profile busy
RS -- remote site standby       SN -- Static Neighbor
LB -- Local site not best-site  RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: customer
  VPLS-id: 601
    Neighbor                  Type  St     Time last up          # Up trans
    10.255.14.217(vpls-id 601) rmt  Up     Oct  8 14:46:54 2012           1
      Remote PE: 10.255.14.217, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 299872
      Negotiated PW status TLV: No
      Local interface: vt-2/0/10.68157697, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls customer neighbor 10.255.14.217 vpls-id 601
```

Meaning

As expected, the VPLS connections are operational on both PE-r devices.

Checking Connectivity

Purpose

Verify that Device CE1 can ping Device CE3.

Action

```
user@CE1> ping 10.255.14.218
PING 10.255.14.218 (10.255.14.218): 56 data bytes
64 bytes from 10.255.14.218: icmp_seq=0 ttl=64 time=0.858 ms
64 bytes from 10.255.14.218: icmp_seq=1 ttl=64 time=0.527 ms
64 bytes from 10.255.14.218: icmp_seq=2 ttl=64 time=0.670 ms
^C
--- 10.255.14.218 ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.527/0.685/0.858/0.136 ms
```

Meaning

The output shows that H-VPLS is operational.

Manually Triggering a Switch from the Active Pseudowire to the Redundant Pseudowire

Purpose

Make sure that the pseudowire between Device PE1 and Device PE2 becomes operational.

Action

```
user@CE1> request l2circuit-switchover virtual-circuit-id 601 neighbor 10.255.14.225
user@CE1> ping 10.255.14.215
PING 10.255.14.215 (10.255.14.215): 56 data bytes
64 bytes from 10.255.14.215: icmp_seq=0 ttl=64 time=0.738 ms
64 bytes from 10.255.14.215: icmp_seq=1 ttl=64 time=0.627 ms
64 bytes from 10.255.14.215: icmp_seq=2 ttl=64 time=0.629 ms
^C
--- 10.255.14.215 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.627/0.665/0.738/0.052 ms
```

Meaning

The successful ping from Device CE1 to Device CE2 shows that the pseudowire between Device PE1 and PE2 is operational. Now, if you ping Device CE3 from Device CE1, the ping should fail.

RELATED DOCUMENTATION

Application Note: Demystifying H-VPLS
Example: Configuring H-VPLS With VLANs | **842**
Redundant Pseudowires for Layer 2 Circuits and VPLS | **224**
Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS | **226**

Configure Hot-Standby Pseudowire Redundancy in H-VPLS

Hot-standby redundant pseudowire provide redundant end-to-end paths on an H-VPLS network. When you enable hot-standby redundancy and the primary pseudowire fails, the network transitions to the redundant pseudowire with minimal disruption.

The benefits of hot-standby pseudowire for H-VPLS include the following:

- Rapid recovery to a rerouted path when the active path has a failure.
- Reduced traffic disruption during the transition.

Figure 63 on page 877 shows a simplified 2-level hierarchical VPLS (H-VPLS) network with an active pseudowire connecting the primary hub HL3-1 to spokes HL4-1 and HL4-2. A standby pseudowire connects the backup hub HL3-2 to spokes HL4-1 and HL4-2. When the primary pseudowire fails, the standby pseudowire takes over.

With the hot-standby enabled, both the primary and backup hub have programmed active and standby pseudowire paths to spoke devices in the Packet Forwarding Engine. Likewise, the spoke devices have programmed active and standby pseudowire paths. Having both active and pseudowire paths preprogrammed on the Packet Forwarding Engine reduces the amount of traffic that gets discarded during transition. Both the active and standby pseudowire can send and receive traffic. So we must give special consideration to preventing traffic loops when spoke devices send unknown traffic. To configure hot-standby in an H-VPLS network, configure the following statements on all hub-and-spoke devices:

On the hub devices:

- Include the `pseudowire-status-tlv` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`* `neighbor` *`address`* `]` hierarchy. PE devices use the pseudowire status type length variable (TLV) to communicate the pseudowire status.
- Include the `hot-standby-vc-on` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`* `neighbor` *`address`* `pseudowire-status-tlv]` hierarchy on the hub devices.

  This allows the backup hub device managing the standby pseudowire to preprogram pseudowire paths to the spoke devices.
- Include the `load-balance per-packet` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy and apply the routing policy to all routes in the forwarding table.

On the spoke devices:

- Include the `pseudowire-status-tlv` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`* `]` hierarchy. PE devices use the pseudowire status type length variable (TLV) to communicate the pseudowire status.

- Include the `hot-standby` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls neighbor` *`address`* `backup-neighbor` *`address`*`]` hierarchy on the spoke devices. Enabling `hot-standby` allows the spoke devices to preprogram the routes and next hops of the standby pseudowire.

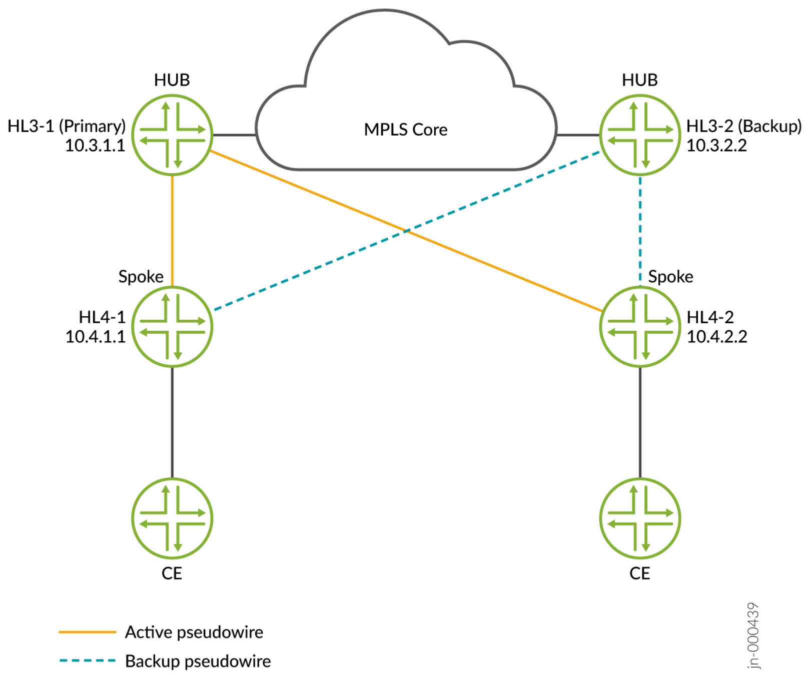
- Include the `vpls-hot-standby-convergence` statement at the `[ routing-options forwarding-table]` hierarchy on the spoke device. When you enable `vpls-hot-standby-convergence`, the Junos device sends advertising messages with different labels on the active and standby pseudowire. The spoke device can then differentiate the return messages coming back from the active and standby pseudowire.

- Include the `load-balance per-packet` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level and apply the routing policy to all routes in the forwarding table.

- (Optional) In some cases, you may want to include a `switchover-delay` at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls mesh-group` *`mesh-group-name`* `neighbor` *`address`* `]` hierarchy on the spoke device. This is the time that the device waits before switching to the backup neighbor after detecting a failure in the primary pseudowire connection. We recommend a delay of 60 seconds.

The following sample hot-standby configuration shows how to configure the hub-and-spoke devices.

Figure 63: H-VPLS Hub-And-Spoke Model



The following is the sample configuration for Hub HL3-1.

```
routing-instances
    LDP-VPLS {
        instance-type vpls;
        interface ge-0/0/0.0;
        protocols {
            vpls {
                no-tunnel-services;
                vpls-id 1;
                neighbor 10.3.2.2;
                mesh-group HL4 {
                    vpls-id 1;
                    local-switching;
                    neighbor 10.4.1.1 {
                        pseudowire-status-tlv hot-standby-vc-on;
                    }
                    neighbor 10.4.2.2 {
                        pseudowire-status-tlv hot-standby-vc-on;
                    }
                }
                connectivity-type permanent;
            }
        }
    }
```

```
policy-options {
    policy-statement PPLB {
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export PPLB;
    }
}
```

The following is the sample configuration for Hub HL3-2.

```
routing-instances
    LDP-VPLS {
        instance-type vpls;
        protocols {
            vpls {
                enable-mac-move-action;
                no-tunnel-services;
                vpls-id 1;
                neighbor 10.1.2.2;
                mesh-group HL4 {
                    vpls-id 1;
                    local-switching;
                    neighbor 10.4.1.1 {
                        pseudowire-status-tlv hot-standby-vc-on;
                    }
                    neighbor 10.4.2.2 {
                        pseudowire-status-tlv hot-standby-vc-on;
                    }
                }
                connectivity-type permanent;
            }
        }
    }
```

```
policy-options {
    policy-statement PPLB {
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export PPLB;
    }
}
```

The following is the sample configuration for the spoke devices (HL4-1 and HL4-2).

```
routing-instances
    LDP-VPLS {
        instance-type vpls;
        interface ge-0/0/0.0;
        protocols {
            vpls {
                no-tunnel-services;
                vpls-id 1;
                neighbor 10.3.1.1 {
                    pseudowire-status-tlv;
                    switchover-delay 60000;
                    revert-time 10;
                    backup-neighbor 10.3.2.2 {
                        hot-standby;
                    }
                }
            }
        }
    }

routing-options {
    forwarding-table {
        vpls-hot-standby-convergence;
    }
}
```

```
policy-options {
    policy-statement PPLB {
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export PPLB;
    }
}
```

Sample Scenario of H-VPLS on ACX Series Routers for IPTV Services

IN THIS SECTION

- Sample Configuration Scenario of H-VPLS for IPTV Services | **881**
- Guidelines for H-VPLS on ACX Routers | **883**

Hierarchical LDP-based VPLS requires a full mesh of tunnel LSPs between all the PE routers that participate in the VPLS service. For each VPLS service, n*(n-1)/2 pseudowires must be set up between the PE routers. Although the full mesh requirement creates signaling overhead, the larger negative impact to large-scale deployment is the packet replication requirements for each provisioned pseudowire on a PE router. Using hierarchical connectivity reduces signaling and replication overhead to facilitate large-scale deployments. .

In a typical IPTV solution, IPTV sources are in the public domain and the subscribers are in the private VPN domain. The objective is to deliver the multicast streams originated from the IPTV source to the set-top boxes or subscribers in the private domain. Generally, for an efficient delivery of multicast data from the IP Sources to the access devices (ACX in this case), P2MP LSPs and mVPN is used. The subscriber devices could then be connected to a VPLS or a Layer 3 VPN domain in Access router and they could be configured to import the multicast routes from an MVPN instance. Because VPLS and MVPN are not supported on ACX routers, an alternative approach can be used to achieve H-VPLS capabilities. The support for PIM snooping in Layer 3 interfaces, IGMP snooping in Layer 2 networks, IRB interfaces, and logical tunnel interfaces enables HVLS support.

An ACX router receives the multicast data on the default VRP context and the data gets forwarded on to the BD through IRB and gets replicated on the BD ports based on the membership detected through IGMP snooping. Unicast control traffic between Subscriber devices and the IPTV subscriber management server goes through the private VPN domain. Aggregation routers have VPLS full-mesh connectivity between each other and ACX works as H-VPLS MTU. There is a PW setup between ACX and the aggregation router. LT interface does the stitching of the Bridge domain with the PW

Sample Configuration Scenario of H-VPLS for IPTV Services

Consider a scenario in which set-top boxes (STBs) or customer premises equipment (CPE) devices are connected to two ACX routers, ACX1 and ACX2. On the ACX routers, a global virtual routing and forwarding (VRF) context, bridge domains, pseudowires, and IRB settings are defined. ACX1 is connected to two MX routers, MX2 and MX3. Connection to ACX1 to MX2 is through an active pseudowire with PIM deployed. PIM is used as the transport protocol in communication between ACX and MX routers. Unicast transmission is also configured. Connection of ACX1 to MX3 is through a backup pseudowire. ACX2 is connected to two MX routers, MX2 and MX3. Connection to ACX2 to MX2 is through an active pseudowire. ACX2 is linked to MX3 using a standby pseudowire. MX2 and MX3 are connected to MX1, which is the root of the LSP. MX1 is linked to an IPTV source.

Point-to-multipoint (P2MP) traffic engineering (TE) LSP is setup from the MX router (MX-1 is connected to the IPTV source) to the aggregation MX routers (MX-2 & MX-3). MX-1 is the root of the LSP. MX-2 and MX-3 are the leaves. Leaves cannot be configured statically or dynamically.

- PIM can be enabled between MX-1 and IPTV Source and MX-1 is the DR for the Multicast source.
- mVPN is configured on the default VRF routing instance. Selective tree can be used for optimal routing.
- PIM-SM is running in the Access network between access routers (ACX-1 and ACX-2) and aggregation routers (MX-1 and MX-2).

  MX-1 and MX-2 are the RPs for the PIM-SM.
- RPF check is disabled for the multicast groups in aggregation routers for it to accept the data from remote source.
- Rendezvous point (RP) redundancy is achieved by configuring auto-RP in ACXs and configuring multiple aggregation routers as RP for every access ring (PIM island).

  ACX has a bridge-domain and subscriber devices are connected to the BD ports. This enables the subscribers to be on the same IP subnet. (This method of configuration suits a topology in which you want all the subscribers connected to the ACXs in a single access ring to be on the same subnet. This makes the DHCP server pool allocation policy easier). No local switching is enabled on the bridge domain, to ensure that subscriber to subscriber communication does not occur locally through the bridge domain.
- Bridge domain has the IRB configuration providing connectivity to the default VRF router instance. IGMP is enabled on the IRB interface. This enables the IGMP join messages send by the subscriber devices to be processed by the routing module and in turn triggers a PIM join towards RP in the default routing instance.
- IGMP snooping is enabled on the bridge domain for an optimal forwarding of multicast data at the BD level.
- ACX router receives the multicast data on the default VRP context and the data gets forwarded on to the BD through IRB and gets replicated on the BD ports based on the membership detected through IGMP snooping.
- Unicast control traffic between Subscriber devices and the IPTV subscriber management server goes through the private VPN domain.

- Aggregation routers have VPLS full-mesh connectivity between each other and ACX works as H-VPLS MTU.
- There is a PW set up between ACX and the aggregation router. LT interface does the stitching of the Bridge domain with the PW.
- An active and a standby pseudowire are set up between ACX and aggregation routers to support redundancy for the control traffic.
- PW terminates into the VPLS instance in the aggregation router. PWs from multiple ACX routers are terminated to the same VPLS instance as all the subscribers across all ACX boxes connected to a same aggregation router are in the same subnet.
- VPLS instance in the aggregation router is connected to the L3VPN instance through IRB interface which has IP address from the same subnet as the subscribers. Subscriber management station that is controlling the subscribers belong to a particular customer is connected to this Layer 3 VPN domain.

Guidelines for H-VPLS on ACX Routers

Keep the following points in mind while configuring H-VPLS on ACX routers:

- Control traffic is limited to 1G or 10G bandwidth based on the logical tunnel (lt-) interface limitation.
- Multicast data delivery is based on PIM on the access network. Therefore, the convergence time is directly related to the convergence time of the IGP protocol configured on the access network.
- Two versions of multicast solutions must be implemented in the provider network for the end-to-end solution to work. Also, you must set up PIM- based solution on the access network and MVPN-based solution on the aggregation or core network. This topology is considered a configuration overhead.
- IPv6 multicast and active-active redundancy models are not supported.
- You can implement an IPTV framework without the MVPN and VPLS support on ACX routers.
- Multicast support using PIM-SM and IGMP is supported,
- Unicast control traffic is restricted to the customer VPN domain.
- Support for multiple subscribers to be on the same IP subnet is provided. Direct communication between the subscribers at the ACX level is disabled.
- IGMP snooping is supported to ensure that the multicast data is not forwarded to the subscribers that are not registered for that data.
- An IRB interface is used for only multicast data delivery from the default VRF context.

RELATED DOCUMENTATION

PIM Overview

Understanding IGMP

Layer 2 Bridge Domains on ACX Series Overview

CHAPTER 36

Configuring Multihoming

IN THIS CHAPTER

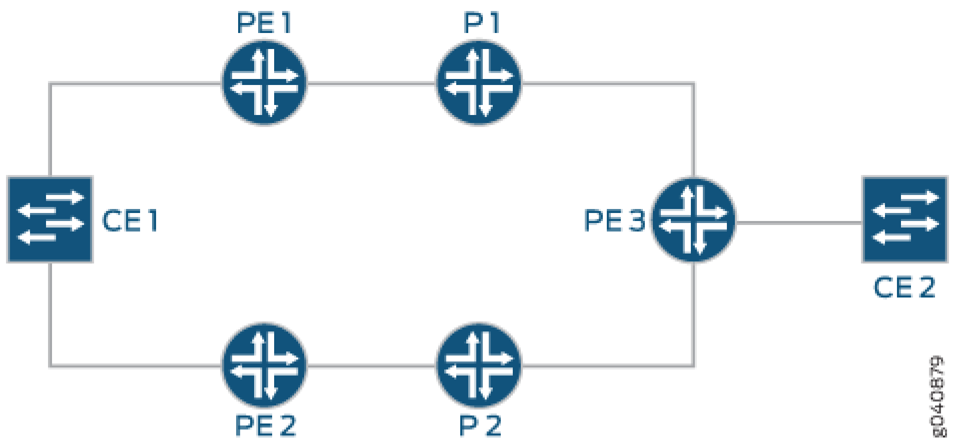
- VPLS Multihoming Overview | 885
- Advantages of Using Autodiscovery for VPLS Multihoming | 888
- Example: Configuring FEC 129 BGP Autodiscovery for VPWS | 889
- Example: Configuring BGP Autodiscovery for LDP VPLS | 913
- Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938
- VPLS Multihoming Reactions to Network Failures | 953
- Configuring VPLS Multihoming | 954
- Example: VPLS Multihoming, Improved Convergence Time | 957
- Example: Configuring VPLS Multihoming (FEC 129) | 974
- Next-Generation VPLS for Multicast with Multihoming Overview | 994
- Example: Next-Generation VPLS for Multicast with Multihoming | 1000

VPLS Multihoming Overview

Virtual private LAN service (VPLS) multihoming enables you to connect a customer site to two or more PE routers to provide redundant connectivity. A redundant PE router can provide network service to the customer site as soon as a failure is detected. VPLS multihoming helps to maintain VPLS service and traffic forwarding to and from the multihomed site in the event of the following types of network failures:

- PE router to CE device link failure
- PE router failure
- MPLS-reachability failure between the local PE router and a remote PE router

Figure 64: CE Device Multihomed to Two PE Routers



NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Figure 64 on page 886 illustrates how a CE device could be multihomed to two PE routers. Device CE1 is multihomed to Routers PE1 and PE2. Device CE2 has two potential paths to reach Device CE1, but only one path is active at any one time. If Router PE1 were the designated VPLS edge (VE) device (also called a designated forwarder), BGP would signal a pseudowire from Router PE3 to Router PE1. If a failure occurred over this path, Router PE2 would be made the designated VE device, and BGP would re-signal the pseudowire from Router PE3 to Router PE2.

Multihomed PE routers advertise network layer reachability information (NLRI) for the multihomed site to the other PE routers in the VPLS network. The NLRI includes the site ID for the multihomed PE routers. For all of the PE routers multihomed to the same CE device, you need to configure the same site ID. The remote VPLS PE routers use the site ID to determine where to forward traffic addressed to the customer site. To avoid route collisions, the site ID shared by the multihomed PE routers must be different than the site IDs configured on the remote PE routers in the VPLS network.

Although you configure the same site ID for each of the PE routers multihomed to the same CE device, you can configure unique values for other parameters, such as the route distinguisher. These values help to determine which multihomed PE router is selected as the designated VE device to be used to reach the customer site.

BEST PRACTICE: We recommend that you configure unique route distinguishers for each multihomed PE router. Configuring unique route distinguishers helps with faster convergence when the connection to a primary multihomed PE router goes down. If you configure unique route distinguishers, the other PE routers in the VPLS network must maintain additional state for the multihomed PE routers.

Remote PE routers in the VPLS network need to determine which of the multihomed PE routers should forward traffic to reach the CE device. To make this determination, remote PE routers use the VPLS path-selection process to select one of the multihomed PE routers based on its NLRI advertisement. Because remote PE routers pick only one of the NLRI advertisements, it establishes a pseudowire to only one of the multihomed PE routers, the PE router that originated the winning advertisement. This prevents multiple paths from being created between sites in the network, preventing the formation of Layer 2 loops. If the selected PE router fails, all PE routers in the network automatically switch to the backup PE router and establish new pseudowires to it.

BEST PRACTICE: To prevent the formation of Layer 2 loops between the CE devices and the multihomed PE routers, we recommend that you employ the Spanning Tree Protocol (STP) on your CE devices. Layer 2 loops can form due to incorrect configuration. Temporary Layer 2 loops can also form during convergence after a change in the network topology.

The PE routers run the BGP path selection procedure on locally originated and received Layer 2 route advertisements to establish that the routes are suitable for advertisement to other peers, such as BGP route reflectors. If a PE router in a VPLS network is also a route reflector, the path selection process for the multihomed site has no effect on the path selection process performed by this PE router for the purpose of reflecting Layer 2 routes. Layer 2 prefixes that have different route distinguishers are considered to have different NLRIs for route reflection. The VPLS path selection process enables the route reflector to reflect all routes that have different route distinguishers to the route reflector clients, even though only one of these routes is used to create the VPLS pseudowire to the multihomed site.

Junos OS supports VPLS multihoming for both BGP VPLS and FEC129 VPLS. See Feature Explorer for information on platform and release support for any VPLS-related feature.

RELATED DOCUMENTATION

Configuring VPLS Multihoming | **954**

Advantages of Using Autodiscovery for VPLS Multihoming | **888**

Example: Configuring VPLS Multihoming (FEC 129) | **977**

Example: VPLS Multihoming, Improved Convergence Time | **957**

Advantages of Using Autodiscovery for VPLS Multihoming

Virtual private LAN service (VPLS) provides a multipoint-to-multipoint Ethernet service that can span one or more metropolitan areas and provides connectivity between multiple sites as if these sites were attached to the same Ethernet LAN.

VPLS uses an IP and MPLS service provider infrastructure. From a service provider’s point of view, use of IP and MPLS routing protocols and procedures instead of the Spanning Tree Protocol (STP), and MPLS labels instead of VLAN IDs, significantly improves the scalability of the VPLS service.

It is frequently a requirement for a service provider to supply its customers with redundant connectivity to one or more sites. This capability is called multihoming.

VPLS multihoming enables you to connect a customer site to two or more PE routers to provide redundant connectivity. A redundant PE router can provide network service to the customer site as soon as a failure is detected. VPLS multihoming helps to maintain VPLS service and traffic forwarding to and from the multihomed site in the event of the following types of network failures:

- PE router to CE device link failure
- PE router failure
- MPLS-reachability failure between the local PE router and a remote PE router

The Junos® operating system (Junos OS) supports both forwarding equivalency class (FEC) 128 and FEC 129. FEC 128 requires manually configured pseudowires. FEC 129 uses VPLS autodiscovery to convey endpoint information. After PE routers are autodiscovered, pseudowires are created automatically.

VPLS multihoming with support for FEC 129 enables you to interoperate with other vendor’s auto-discovery for LDP-signaled VPLS. This interoperability allows you to select the vendor that offers the best value.

RELATED DOCUMENTATION

VPLS Multihoming Overview | 885
Example: Configuring VPLS Multihoming (FEC 129) | 974

Example: Configuring FEC 129 BGP Autodiscovery for VPWS

IN THIS SECTION

- Understanding VPWS | 889
- Understanding FEC 129 BGP Autodiscovery for VPWS | 892
- Example: Configuring FEC 129 BGP Autodiscovery for VPWS | 895

Understanding VPWS

IN THIS SECTION

- Supported and Unsupported Features | 891

Virtual private wire service (VPWS) Layer 2 VPNs employ Layer 2 services over MPLS to build a topology of point-to-point connections that connect end customer sites in a VPN. These Layer 2 VPNs provide an alternative to private networks that have been provisioned by means of dedicated leased lines or by means of Layer 2 virtual circuits. The service provisioned with these Layer 2 VPNs is known as VPWS. You configure a VPWS *instance* on each associated edge device for each VPWS Layer 2 VPN.

Traditional VPNs over Layer 2 circuits require the provisioning and maintenance of separate networks for IP and for VPN services. In contrast, *VPWS* enables the sharing of a provider's core network infrastructure between IP and Layer 2 VPN services, reducing the cost of providing those services.

Junos OS supports two types of VPWS Layer 2 VPNs:

- Kompella Layer 2 VPNs, which use BGP for autodiscovery and signaling.
- FEC 129 BGP autodiscovery for VPWS, which uses BGP for autodiscovery and LDP as the signaling protocol.

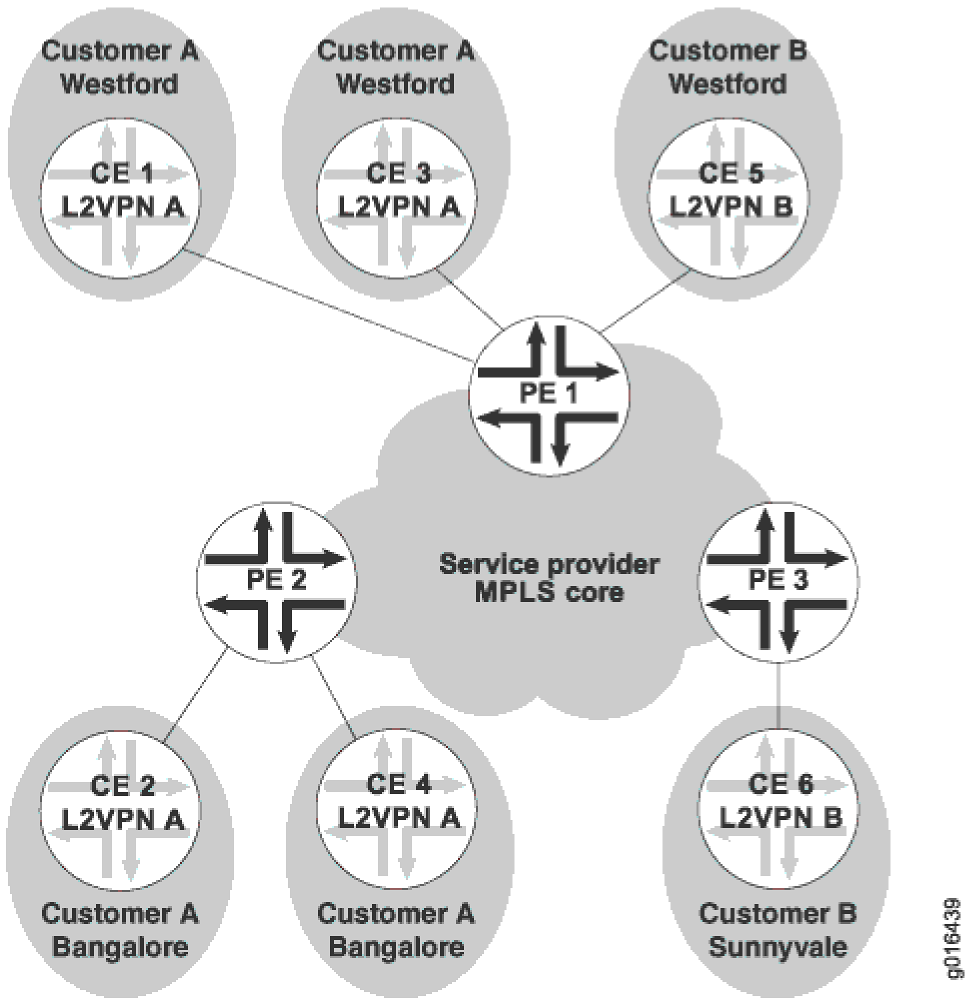
FEC 129 BGP autodiscovery for VPWS requires the `l2vpn-id`, `source-attachment-identifier`, and `target-attachment-identifier` statements. Kompella Layer 2 VPNs require the `site-identifier` and `remote-site-id` statements.

NOTE: VPWS creates pseudowires that emulate Layer 2 circuits. A virtual private LAN service (VPLS) network is similar to VPWS, but provides point-to-multipoint traffic forwarding in contrast to the VPWS Layer 2 VPN's point-to-point traffic forwarding. If you need point-to-multipoint service instead of point-to-point service, consider using VPLS instead of VPWS.

A VPWS Layer 2 VPN can have either a full-mesh or a hub-and-spoke topology. The tunneling mechanism in the core network typically is MPLS. However, VPWS can also use other tunneling protocols, such as GRE. VPWS is similar to Martini Layer 2 services over MPLS, and employs a similar encapsulation scheme for forwarding traffic.

Figure 65 on page 890 illustrates an example of a simple VPWS Layer 2 VPN topology.

Figure 65: VPWS Sample Topology



In this example, the service provider offers VPWS services to Customer A and Customer B. Customer A wants to create a full mesh of point-to-point links between Westford and Bengaluru. Customer B needs only a single point-to-point link between Westford and Sunnyvale. The service provider uses BGP and MPLS signaling in the core, and creates a set of unidirectional pseudowires at each provider edge (PE) device to separately cross-connect each customer's Layer 2 circuits.

In order to provision this service, the provider configures two VPWS Layer 2 VPNs, Layer 2 VPN A and Layer 2 VPN B. The circuit cross-connect (CCC) encapsulation type (`ethernet-ccc` or `vlan-ccc`) is configured for each VPWS Layer 2 VPN. All interfaces in a given VPWS Layer 2 VPN must be configured with the VPWS Layer 2 VPN's encapsulation type.

Local and remote site information for the interfaces identifies the cross-connect. Local cross-connects are supported when the interfaces that are connected belong to two different sites configured in the same VPWS instance and on the same PE device.

BGP advertises reachability for the VPNs. The BGP configuration is similar to that used for other VPN services, such as Layer 3 VPNs and VPLS. MPLS is configured to set up base LSPs to the remote PE devices similarly to the other VPN services.

Junos OS provides VPWS support the following configuration methods:

- Pseudowires are manually configured using Forwarding Equivalence Class (FEC) 128.
- Pseudowires are signaled by LDP using FEC 129. This arrangement reduces the configuration burden that is associated with statically configured Layer 2 circuits while still using LDP as the underlying signaling protocol.

Supported and Unsupported Features

Junos OS supports the following features with VPWS :

- Intra-AS VPWS functionality using BGP for autodiscovery and FEC 129 LDP for pseudowire signaling.
- *Graceful Routing Engine switchover*.
- Operation, administration, and maintenance (OAM) mechanisms, including Bidirectional Forwarding Detection and MPLS ping.
- FEC 128 LDP signaling with static configuration (in Junos OS this is configured within `protocols l2circuit`). With this option, there is no BGP autodiscovery.

Junos OS does not support the following VPWS functionality:

- Multihoming of customer sites to multiple PE devices using the BGP site model of multihoming.
- Terminating FEC 129 VPWS into a mesh group of an FEC 129 VPLS instance.

- Intra-AS VPWS functionality using BGP for autodiscovery and FEC 128 LDP for pseudowire signaling.
- FEC 129 VPWS without BGP autodiscovery.
- Static configuration of VPWS with FEC 129 signaling.
- *Nonstop active routing*.
- Multi-segment pseudowires.
- Interworking of FEC 128 and FEC 129 VPWS.
- Statically configured Layer 2 circuit-style pseudowire redundancy.
- Inter-AS deployments.

SEE ALSO

Example: Configuring BGP Autodiscovery for LDP VPLS | **913**

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | **938**

Understanding FEC 129 BGP Autodiscovery for VPWS

IN THIS SECTION

- Supported Standards in FEC 129 BGP Autodiscovery for VPWS | **893**
- Routes and Routing Table Interaction in FEC 129 BGP Autodiscovery for VPWS | **893**
- Layer 2 VPN Behavior in FEC 129 BGP Autodiscovery for VPWS | **894**
- BGP Autodiscovery Behavior in FEC 129 BGP Autodiscovery for VPWS | **894**
- LDP Signaling Behavior in VPWS in FEC 129 BGP Autodiscovery for VPWS | **895**

The major functional components in a *VPWS* with FEC 129 are BGP, LDP, and the Layer 2 VPN module of Junos OS. BGP is responsible for distributing the local autodiscovery routes created on each PE device to all other PE devices. LDP is responsible for using the autodiscovery information provided by BGP to set up targeted LDP sessions over which to signal the pseudowires. The Layer 2 VPN is the glue that binds the BGP and LDP functionalities together.

Supported Standards in FEC 129 BGP Autodiscovery for VPWS

The relevant RFCs for this feature are as follows:

- RFC 4447, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*
- RFC 6074, *Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*

Routes and Routing Table Interaction in FEC 129 BGP Autodiscovery for VPWS

BGP, LDP, and Layer 2 VPNs interact through different types of routes installed in the *instance*.l2vpn.0 table. Theroutes that are present in the table are autodiscovery routes and pseudowire routes.

- Autodiscovery routes are used by BGP to allow autodiscovery of remote source access individual identifiers (SAIIs) (the sources of the point-to-point pseudowires) and PE device addresses. Autodiscovery routes are advertised when you configure the `l2vpn auto-discovery-only` address family.

  The format of the autodiscovery routes is a combination of the route distinguisher and the SAII. For example: 10.255.0.1:100:0.0.0.1/96 AD.

  Table 21 on page 893 lists the route elements and the number of associated bytes allocated to each element.

  Table 21: Autodiscovery Route Format

Route Element	Bytes
RD	8 bytes
SAII	4 bytes

  The `l2vpn-id` of the FEC 129 VPWS instance is attached to the route in a BGP extended community. One autodiscovery route is advertised for each source attachment identifier (SAI) in the instance.

- Pseudowire routes are installed by the Layer 2 VPN (local) and LDP (remote) to represent the bidirectional components of the pseudowire. For example: NoCtrlWord:5:100:200:2:0.0.0.1/176. The format of the routes is described in Table 22 on page 894.

Table 22: Pseudowire Route Format

Field Name	Field Description
Pseudowire type + control word bit	2 bytes
Remote PE address	4 bytes
Attachment group identifier (AGI) The AGI field of the pseudowire route is always set to the `l2vpn-id` of the instance.	8 bytes
SAII	4 bytes
Target attachment individual identifier (TAII)	4 bytes

Layer 2 VPN Behavior in FEC 129 BGP Autodiscovery for VPWS

A Layer 2 VPN installs a locally generated autodiscovery route into the instance.l2vpn.0 table for every SAII configured in an FEC 129 VPWS instance. The extended community containing the `l2vpn-id` is attached when the route is added to the instance.l2vpn.0 table.

For each autodiscovered SAII from a remote neighbor where the `l2vpn-id` matches the local `l2vpn-id` and the received SAII matches a locally configured TAII, the Layer 2 VPN obtains an MPLS label and generates a pseudowire route and adds it to the instance.l2vpn.0 table. The remote PE address is copied from the BGP protocol next hop for the autodiscovery route.

The Layer 2 VPN module of Junos OS is responsible for installing the forwarding routes into the mpls.0 table as usual.

BGP Autodiscovery Behavior in FEC 129 BGP Autodiscovery for VPWS

Local autodiscovery routes installed by the Layer 2 VPN in the instance.l2vpn.0 table are advertised by BGP to remote PE devices s`l2vpn auto-discovery-only` address family according to the instance and BGP export policies.

On the receiving side, BGP accepts autodiscovery routes from remote peers and installs them in the local bgp.l2vpn.0 table, if they are allowed by inbound policy. The route is installed, and a secondary route is imported into the instance.l2vpn.0 table when an import route target match between the route and instance is found.

LDP Signaling Behavior in VPWS in FEC 129 BGP Autodiscovery for VPWS

In the Junos OS implementation of LDP, the router monitors for routes from instance.l2vpn.0 for any instance configured for FEC 129 VPWS. These routes are identified by the `instance-type l2vpn` statement in the routing instance and the presence of the `l2vpn-id` statement.

When a BGP autodiscovery route is installed, LDP sets up a targeted session with the remote peer, where the peer address is identified as the protocol next hop of the BGP autodiscovery route.

When a pseudowire route is installed in the instance.l2vpn.0 table, LDP uses the parameters associated with the route to signal the creation of the pseudowire using FEC 129. Upon receiving an FEC 129 label mapping message from a remote peer, LDP installs the pseudowire route in the ldp.l2vpn.0 table.

Upon a successful `l2vpn-id` match with a configured FEC 129 VPWS instance, a secondary pseudowire route is imported to the instance.l2vpn.0 table. If an outgoing pseudowire has not already been set up when the incoming pseudowire signaling is received, LDP initiates the outgoing pseudowire creation as well.

SEE ALSO

Understanding VPWS | 491

Example: Configuring FEC 129 BGP Autodiscovery for VPWS | 501

Example: Configuring FEC 129 BGP Autodiscovery for VPWS

IN THIS SECTION

- Requirements | 896
- Overview | 896
- Configuration | 901
- Verification | 907

This example shows how to configure the virtual private wire service (VPWS), where remote provider edge (PE) devices are automatically discovered dynamically by BGP, and pseudowires are signaled by LDP using FEC 129. This arrangement reduces the configuration burden that is associated with statically configured Layer 2 circuits while still using LDP as the underlying signaling protocol.

Requirements

This example requires Junos OS Release 13.2 or later on the PE devices.

Overview

IN THIS SECTION

- Topology Diagram | 900

Because VPWS is a point-to-point service, FEC 129 VPWS routing instances are configured as `instance-type l2vpn`. As with FEC 129 VPLS, FEC 129 VPWS uses the `l2vpn-id` statement to define the Layer 2 VPN of which the routing instance is a member. The presence of the `l2vpn-id` statement designates that FEC 129 LDP signaling is used for the routing instance. The absence of `l2vpn-id` indicates that BGP signaling is used instead.

The point-to-point nature of VPWS requires that you specify the source access individual identifier (SAII) and the target access individual identifier (TAII). This SAII-TAII pair defines a unique pseudowire between two PE devices.

The SAII is specified with the `source-attachment-identifier` statement within the FEC 129 VPWS routing instance. You configure the source attachment identifier and the interfaces to associate with that source attachment identifier. Under each interface, you can configure the TAII with the `target-attachment-identifier` statement. If the configured target identifier matches a source identifier advertised by a remote PE device by way of a BGP autodiscovery message, the pseudowire between that source-target pair is signaled. If there is no match between an advertised source identifier and the configured target identifier, the pseudowire is not established.

Sample: VPWS Configuration with Multiple Interfaces and Sites

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
        route-distinguisher 10.255.0.1:200;
        l2vpn-id l2vpn-id:100:200;
        vrf-target target:100:200;
        protocols l2vpn {
            site CUSTOMER-1 {
                source-attachment-identifier 1;
                interface ge-0/0/1.0 {
                    target-attachment-identifier 2;
                }
                interface ge-0/0/2.0 {
                    target-attachment-identifier 3;
                }
            }
        }
    }
}
```

You can configure multiple interfaces within a site, because each SAII-TAII pair defines a unique pseudowire, as shown with pseudowires 1-2 and 1-3 in the sample configuration. Both the source and target access identifiers are 4-byte numbers and can only be configured in FEC 129 VPWS instances where the `instance-type` is `l2vpn` and the `l2vpn-id` configuration statement is present.

You can specify the source and target identifiers as plain unsigned integers in the range 1 through 4,292,967,295.

The Layer 2 circuit and Layer 2 VPN services allow many optional parameters to be included on a per-pseudowire basis. FEC 129 VPWS allows such parameters as MTU settings, community tagging, and inclusion of a control word, as shown in this sample configuration:

Sample: VPWS Configuration with Optional Configuration Parameters

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        interface ge-0/0/2.0;
        interface ge-0/0/3.0;
        route-distinguisher 10.255.0.1:200;
        l2vpn-id l2vpn-id:100:200;
        vrf-target target:100:200;
        protocols l2vpn {
            site CUSTOMER-1 {
                source-attachment-identifier 1;
                community COMM;
                control-word ;
                encapsulation-type ethernet;
                ignore-encapsulation-mismatch;
                ignore-mtu-mismatch;
                mtu 1500;
                no-control-word;
                interface ge-0/0/1.0 {
                    target-attachment-identifier 2;
                }
                interface ge-0/0/2.0 {
                    target-attachment-identifier 3;
                    community COMM;
                    control-word;
                    encapsulation-type ethernet;
                    ignore-encapsulation-mismatch;
                    ignore-mtu-mismatch;
                    mtu 1500;
                    no-control-word;
                }
            }
        }
    }
}
```

When configured within the site, the defined parameters affect any pseudowire originating from that site. When configured under an interface, the defined parameters affect that single specific pseudowire. This allows you to manipulate the parameters across all pseudowires associated with a particular local site in one place in the configuration.

Like other point-to-point services, the interfaces configured as members of the FEC 129 VPWS instance must be configured for CCC encapsulation and the CCC address family, as shown here:

```
interfaces {
    ge-0/0/1 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/2 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    ge-0/0/3 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
}
```

You can use `vlan-ccc` instead of `ethernet-ccc`.

To support the basic FEC 129 VPWS functionality, the BGP sessions on the PE devices also need to be configured with the BGP `auto-discovery-only` address family to allow exchange of the autodiscovery routes. If traditional BGP VPLS or Layer 2 VPN service is also provisioned on the PE devices, the address family `l2vpn signaling` is also required, as shown here:

```
bgp {
    group pe {
        type internal;
        local-address 10.255.0.1;
        family l2vpn {
            auto-discovery-only;
            signaling;
        }
        neighbor 10.255.0.2;
        neighbor 10.255.0.3;
    }
}
```

The following configuration sample shows an FEC 129 VPWS routing instance with the operation, administration, and maintenance (OAM) (ping and BFD) configuration options:

Sample: VPWS Configuration with OAM

```
routing-instances {
    FEC129-VPWS {
        instance-type l2vpn;
        interface ge-0/0/1.0;
        route-distinguisher 10.255.0.1:200;
        l2vpn-id l2vpn-id:100:200;
        vrf-target target:100:200;
        protocols l2vpn {
            oam {
                ping-interval 600;
                bfd-liveness-detection {
                    minimum-interval 200;
                }
            }
            site CUSTOMER {
                source-attachment-identifier 1;
                oam {
                    ping-interval 600;
                    bfd-liveness-detection {
                        minimum-interval 200;
                    }
                }
                interface ge-0/0/1.0 {
                    oam {
                        ping-interval 600;
                        bfd-liveness-detection {
                            minimum-interval 200;
                        }
                    }
                    target-attachment-identifier 2;
                }
            }
        }
    }
}
```

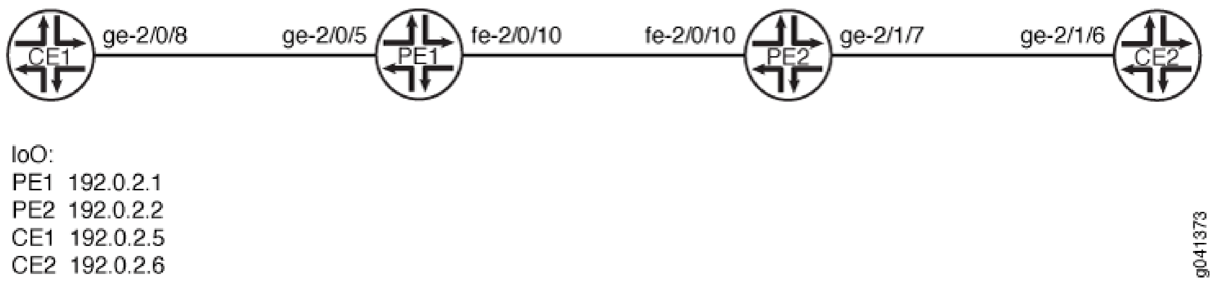
OAM options configured under `protocols l2vpn` apply to all sites and pseudowires in the routing instance. OAM options configured under a particular site apply to the pseudowires configured under that site. OAM options configured under a particular interface apply to the pseudowire configured under that interface.

Topology Diagram

Figure 66 on page 901 shows the topology used in this example.

This example uses a simple topology with two PE devices and two customer edge (CE) devices.

Figure 66: Simple VPWS Topology



"CLI Quick Configuration" on page 901 shows the configuration for all of the devices in Figure 66 on page 901. The section "Step-by-Step Procedure" on page 903 describes the steps on Device PE1 and Device PE2.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **901**
- Procedure | **903**

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 CE1

```
set interfaces ge-2/0/8 unit 0 description CE1_to_PE1
set interfaces ge-2/0/8 unit 0 family inet address 172.16.0.1/24
set interfaces lo0 unit 0 family inet address 192.0.2.5/24
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 CE2

```
set interfaces ge-2/1/6 unit 0 description CE2_to_PE2
set interfaces ge-2/1/6 unit 0 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 192.0.2.6/24
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/6.0
```

Device PE1

```
set interfaces ge-2/0/5 encapsulation ethernet-ccc
set interfaces ge-2/0/5 unit 0 description PE1_to_CE1
set interfaces ge-2/0/5 unit 0 family ccc
set interfaces fe-2/0/10 unit 0 description to_PE2
set interfaces fe-2/0/10 unit 0 family inet address 10.0.0.1/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.1/24
set protocols mpls interface fe-2/0/10.0
set protocols bgp local-address 192.0.2.1
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances FEC129-VPWS instance-type l2vpn
set routing-instances FEC129-VPWS interface ge-2/0/5.0
set routing-instances FEC129-VPWS route-distinguisher 192.0.2.1:100
set routing-instances FEC129-VPWS l2vpn-id l2vpn-id:100:100
set routing-instances FEC129-VPWS vrf-target target:100:100
set routing-instances FEC129-VPWS protocols l2vpn site ONE source-attachment-identifier 1
set routing-instances FEC129-VPWS protocols l2vpn site ONE interface ge-2/0/5.0 target-
attachment-identifier 2
set routing-options autonomous-system 64510
```

Device PE2

```
set interfaces ge-2/1/7 encapsulation ethernet-ccc
set interfaces ge-2/1/7 unit 0 description PE2_to_CE2
set interfaces ge-2/1/7 unit 0 family ccc
set interfaces fe-2/0/10 unit 0 description to_PE1
set interfaces fe-2/0/10 unit 0 family inet address 10.0.0.2/30
set interfaces fe-2/0/10 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.2/24
set protocols mpls interface fe-2/0/10.0
set protocols bgp local-address 192.0.2.2
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.1
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface fe-2/0/10.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface fe-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances FEC129-VPWS instance-type l2vpn
set routing-instances FEC129-VPWS interface ge-2/1/7.0
set routing-instances FEC129-VPWS route-distinguisher 192.0.2.2:100
set routing-instances FEC129-VPWS l2vpn-id l2vpn-id:100:100
set routing-instances FEC129-VPWS vrf-target target:100:100
set routing-instances FEC129-VPWS protocols l2vpn site TWO source-attachment-identifier 2
set routing-instances FEC129-VPWS protocols l2vpn site TWO interface ge-2/1/7.0 target-
attachment-identifier 1
set routing-options autonomous-system 64510
```

Procedure

Step-by-Step Procedure

To configure a FEC 129 VPWS:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-2/0/5 encapsulation ethernet-ccc
user@PE1# set ge-2/0/5 unit 0 description PE1_to_CE1
```

```
user@PE1# set ge-2/0/5 unit 0 family ccc
user@PE1# set fe-2/0/10 unit 0 description to_PE2
user@PE1# set fe-2/0/10 unit 0 family inet address 10.0.0.1/30
user@PE1# set fe-2/0/10 unit 0 family mpls
user@PE1# set lo0 unit 0 family inet address 192.0.2.1/24
```

2. Configure MPLS on the core-facing interface.

```
[edit protocols mpls]
user@PE1# set interface fe-2/0/10.0
```

3. Configure BGP.

```
[edit protocols bgp]
user@PE1# set local-address 192.0.2.1
user@PE1# set group pe-pe type internal
user@PE1# set group pe-pe family l2vpn auto-discovery-only
user@PE1# set group pe-pe family l2vpn signaling
user@PE1# set group pe-pe neighbor 192.0.2.2
```

4. Configure an interior gateway protocol, such as IS-IS or OSPF.

   If you use OSPF, enable traffic engineering. Traffic engineering is supported by IS-IS by default.

```
[edit protocols ospf]
user@PE1# set traffic-engineering
user@PE1# set area 0.0.0.0 interface lo0.0 passive
user@PE1# set area 0.0.0.0 interface fe-2/0/10.0
```

5. Configure LDP on the core-facing interface and on the loopback interface.

```
[edit protocols ldp]
user@PE1# set interface fe-2/0/10.0
user@PE1# set interface lo0.0
```

6. Configure the VPWS routing instance.

LDP listens for routes from instance.l2vpn.0 for any instance configured for FEC 129 VPWS. These routes are identified by the `instance-type l2vpn` statement in the routing instance and the presence of the `l2vpn-id` statement.

Make sure that the `target-attachment-identifier` matches the `source-attachment-identifier` in the remote PE device's corresponding site. In this example, the pseudowire is established between Device PE1 and Device PE2. Device PE1 uses SAI 1 and TAI 2, while Device PE2 uses the opposite, SAI 2 and TAI 1.

```
[edit routing-instances FEC129-VPWS]
user@PE1# set instance-type l2vpn
user@PE1# set interface ge-2/0/5.0
user@PE1# set route-distinguisher 192.0.2.1:100
user@PE1# set l2vpn-id l2vpn-id:100:100
user@PE1# set vrf-target target:100:100
user@PE1# set protocols l2vpn site ONE source-attachment-identifier 1
user@PE1# set protocols l2vpn site ONE interface ge-2/0/5.0 target-attachment-identifier 2
```

7. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@PE1# set autonomous-system 64510
```

8. 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 interfaces`, `show protocols`, `show routing-instances`, and `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show interfaces
ge-2/0/5 {
    encapsulation ethernet-ccc;
    unit 0 {
        description PE1_to_CE1;
        family ccc;
    }
}
fe-2/0/10 {
    unit 1 {
        description to_PE2;
        family inet {
            address 10.0.0.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface fe-2/0/10.0;
}
bgp {
    local-address 192.0.2.1;
    group pe-pe {
        type internal;
        family l2vpn {
            auto-discovery-only;
            inactive: signaling;
        }
        neighbor 192.0.2.2;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-2/0/10.0;
    }
}
ldp {
    interface fe-2/0/10.0;
    interface lo0.0;
}
```

```
user@PE1# show routing-instances
FEC129-VPWS {
    instance-type l2vpn;
    interface ge-2/0/5.0;
    route-distinguisher 192.0.2.1:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        l2vpn {
            site ONE {
                source-attachment-identifier 1;
                interface ge-2/0/5.0 {
                    target-attachment-identifier 2;
                }
            }
        }
    }
}
```

```
user@PE1# show routing-options
autonomous-system 64510;
```

Verification

IN THIS SECTION

- Verifying the Routes | **908**
- Checking Connectivity Between the CE Devices | **910**
- Checking the VPWS Connections | **910**

- Checking Connectivity Between the PE Devices | **912**

Confirm that the configuration is working properly.

Verifying the Routes

Purpose

Verify that the expected routes are learned.

Action

From operational mode, enter the `show route` command.

```
user@PE1> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.1/24         *[Direct/0] 6d 21:16:32
                    > via lo0.0
192.0.2.2/24         *[OSPF/10] 6d 21:15:31, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0
10.0.0.0/30        *[Direct/0] 6d 21:16:31
                    > via fe-2/0/10.0
10.0.0.1/32        *[Local/0] 6d 21:16:32
                      Local via fe-2/0/10.0
203.0.113.0/24       *[OSPF/10] 6d 21:16:34, metric 1
                      MultiRecv

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2/24         *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0

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

```
0                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
1                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
2                  *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
13                 *[MPLS/0] 6d 21:16:33, metric 1
                      Receive
299808             *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0, Pop
299808(S=0)        *[LDP/9] 5d 22:25:19, metric 1
                    > to 10.0.0.2 via fe-2/0/10.0, Pop
299824             *[L2VPN/7] 5d 22:25:18
                    > via ge-2/0/5.0, Pop
ge-2/0/5.0         *[L2VPN/7] 5d 22:13:02, metric2 1
                    > to 10.0.0.2 via fe-2/0/10.0, Push 299872

bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2:100:0.0.0.2/96 AD
                   *[BGP/170] 6d 20:51:23, localpref 100, from 192.0.2.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-2/0/10.0

ldp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2:NoCtrlWord:5:100:100:0.0.0.2:0.0.0.1/176
                   *[LDP/9] 5d 22:13:02
                      Discard

FEC129-VPWS.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.1:100:0.0.0.1/96 AD
                   *[L2VPN/170] 6d 20:53:26, metric2 1
                      Indirect
192.0.2.2:100:0.0.0.2/96 AD
                   *[BGP/170] 6d 20:51:23, localpref 100, from 192.0.2.2
                      AS path: I, validation-state: unverified
                    > to 10.0.0.2 via fe-2/0/10.0
192.0.2.2:NoCtrlWord:5:100:100:0.0.0.1:0.0.0.2/176
```

```
                   *[L2VPN/7] 6d 20:51:23, metric2 1
                    > to 10.0.0.2 via fe-2/0/10.0
192.0.2.2:NoCtrlWord:5:100:100:0.0.0.2:0.0.0.1/176
                   *[LDP/9] 5d 22:13:02
                      Discard
```

Meaning

The output shows all the learned routes, including the autodiscovery (AD) routes.

Checking Connectivity Between the CE Devices

Purpose

Verify that Device CE1 can ping Device CE2.

Action

```
user@CE1> ping 192.0.2.6
PING 192.0.2.6 (192.0.2.6): 56 data bytes
64 bytes from 192.0.2.6: icmp_seq=0 ttl=64 time=0.679 ms
64 bytes from 192.0.2.6: icmp_seq=1 ttl=64 time=0.524 ms
^C
--- 192.0.2.6 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.524/0.602/0.679/0.078 ms
```

Meaning

The output shows that the VPWS is operational.

Checking the VPWS Connections

Purpose

Make sure that all of the FEC 129 VPWS connections come up correctly.

Action

```
user@PE1> show l2vpn connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: FEC129-VPWS
  L2vpn-id: 100:100
  Local source-attachment-id: 1 (ONE)
    Target-attachment-id      Type  St     Time last up          # Up trans
    2                         rmt   Up     Nov 28 16:16:14 2012           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 299792, Outgoing label: 299792
      Local interface: ge-2/0/5.0, Status: Up, Encapsulation: ETHERNET
```

Meaning

As expected, the connection is up. The output includes the source attachment ID and the target attachment ID.

Checking Connectivity Between the PE Devices

Purpose

Verify that Device PE1 can ping Device PE2. The `ping mpls l2vpn fec129` command accepts SAIs and TAIs as integers or IP addresses and also allows you to use the CE-facing interface instead of the other parameters (`instance`, `local-id`, `remote-id`, `remote-pe-address`).

Action

```
user@PE1> ping mpls l2vpn fec129 instance FEC129-VPWS remote-id 2 remote-pe-address 192.0.2.2
local-id 1
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

```
user@PE1> ping mpls l2vpn fec129 interface ge-2/0/5.0
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Meaning

The output shows that the VPWS is operational.

SEE ALSO

Example: Configuring BGP Autodiscovery for LDP VPLS | 913

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | 913
Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938
Example: Configuring VPLS Multihoming (FEC 129) | 974
Configuring the Local Site on PE Routers in Layer 2 VPNs | 150

Example: Configuring BGP Autodiscovery for LDP VPLS

IN THIS SECTION

● Requirements | 913
● Overview | 914
● Configuration | 916
● Verification | 937

This example describes how to configure BGP autodiscovery for LDP VPLS, as specified in forwarding equivalency class (FEC) 129. FEC 129 uses BGP autodiscovery to convey endpoint information, so you do not need to manually configure pseudowires.

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms
- Junos OS Release 10.4R2 or later

If you are using M Series or T Series routers, the PE routers must have either virtual loopback tunnel (vt) interfaces or label-switched interfaces (LSIs). On M Series and T Series routers, VPLS uses tunnel-based PICs to create virtual ports on vt interfaces. If you do not have a tunnel-based PIC installed on your M Series or T Series router, you can still configure VPLS by using LSIs to support the virtual ports. Use of LSIs requires Ethernet-based PICs installed in an Enhanced Flexible PIC Concentrator (FPC).

You do not need to use routers for the CE devices. For example, the CE devices can be EX Series Ethernet Switches.

Overview

IN THIS SECTION

- Topology Diagram | **915**

All PE routers in a VPLS network operate like a large, distributed Ethernet switch to provide Layer 2 services to attached devices. This example shows a minimum configuration for PE routers and CE devices to create an autodiscovered VPLS network. The topology consists of five routers: two PE routers, two CE routers, and an optional route reflector (RR). The PE routers use BGP to autodiscover two different VPLS instances that are configured on both PE routers. Then the PE routers use LDP to automatically signal two pseudowires between the discovered end points. Finally, the PE routers bring up both VPLS instances for forwarding traffic. Each CE device is configured with two VLANs, with each VLAN belonging to different VPLS instances in the PE routers.

This example includes the following settings:

- `auto-discovery-only`—Allows the router to process only the autodiscovery network layer reachability information (NLRI) update messages for LDP-based Layer 2 VPN and VPLS update messages (BGP_L2VPN_AD_NLRI) (FEC 129). Specifically, the `auto-discovery-only` statement notifies the routing process (rpd) to expect autodiscovery-related NLRI messages so that information can be deciphered and used by LDP and VPLS. You can configure this statement at the global, group, and neighbor levels for BGP. The `auto-discovery-only` statement must be configured on all PE routers in the VPLS. If you configure route reflection, the `auto-discovery-only` statement is also required on P routers that act as the route reflector in supporting FEC 129-related updates.

  The `signaling` statement is not included in this example but is discussed here for completeness. The `signaling` statement allows the router to process only the BGP_L2VPN_NLRIs used for BGP-based Layer 2 VPNs (FEC 128).

  For interoperation scenarios in which a PE router must support both types of NLRI (FEC 128 and FEC 129), you can configure both the `signaling` statement and the `auto-discovery-only` statement. For example, a single PE router might need to process a combination of BGP-signaled virtual private wire service (VPWS) and LDP-signaled VPLS assisted by BGP autodiscovery. Configuring both the `signaling` statement and the `auto-discovery-only` statement together allows both types of signaling to run independently. The `signaling` statement is supported at the same hierarchy levels as the `auto-discovery-only` statement.

- `cluster`—Configuring a route reflector is optional for FEC 129 autodiscovered PE routers. In this example, the `cluster` statement configures Router RR to be a route reflector in the IBGP group. For inbound updates, BGP autodiscovery NLRI messages are accepted if the router is configured to be a route reflector or if the `keep all` statement is configured in the IBGP group.

- `l2vpn-id`—Specifies a globally unique Layer 2 VPN community identifier for the instance. This statement is configurable for routing instances of type `vpls`.

  You can configure the following formats for the community identifier:

  - Autonomous system (AS) number format—`l2vpn-id:`*`as-number`*`:`*`2-byte-number`* . For example: `l2vpn-id:100:200`. The AS number can be in the range from 1 through 65,535.

  - IPv4 format—`l2vpn-id:`*`ip-address`*`:`*`2-byte-number`* . For example: `l2vpn-id:10.1.1.1:2`.

- `vrf-target`—Defines the import and export route targets for the NLRI. You must either configure the `vrf-target` statement or the `vrf-import` and `vrf-export` statements to define the instance import and export policy or the import and export route targets for the NLRI. This example uses the `vrf-target` statement.

- `route-distinguisher`—Forms part of the BGP autodiscovery NLRI and distinguishes to which VPN or VPLS routing instance each route belongs. Each route distinguisher is a 6-byte value. You must configure a unique route distinguisher for each routing instance.

  You can configure the following formats for the route distinguisher:

  - AS number format— *`as-number`*`:`*`2-byte-number`*

  - IPv4 format— *`ip-address`*`:`*`2-byte-number`*

Two notable statements are included in this example. These statements are important for interoperability with other vendors' equipment. The interoperability statements are not necessary for the topology that is used in this example, but they are included for completeness.

The interoperability statements are as follows:

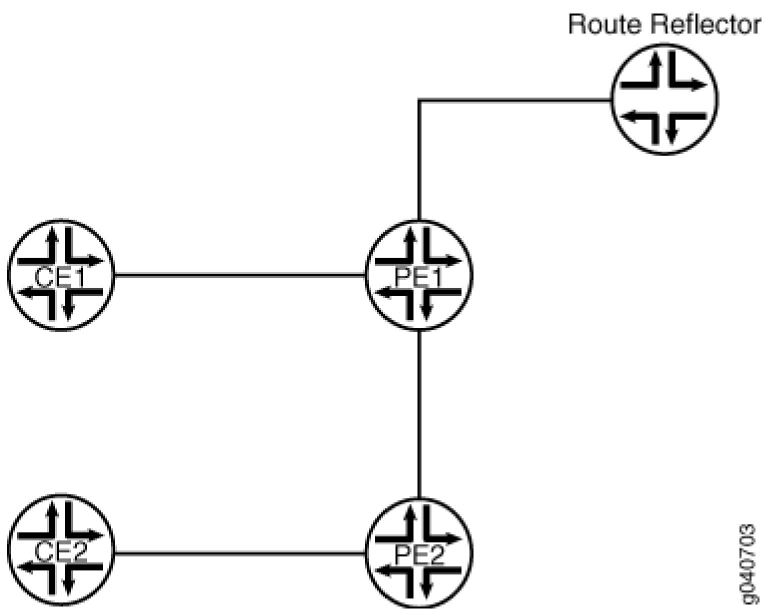
- `input-vlan-map pop`—Removes an outer VLAN tag from the top of the VLAN tag stack.

- `output-vlan-map push`—Adds an outer VLAN tag in front of the existing VLAN tag.

Topology Diagram

Figure 67 on page 916 shows the topology used in this example.

Figure 67: BGP Autodiscovery for LDP VPLS



Configuration

IN THIS SECTION

- CLI Quick Configuration | **916**
- Router PE1 | **920**
- Device CE1 | **925**
- Router PE2 | **927**
- Device CE2 | **932**
- Router RR | **934**

CLI Quick Configuration

To quickly configure BGP autodiscovery for LDP VPLS, copy the following commands, remove any line breaks, and then paste the commands into the CLI of each device.

On Router PE1:

```
[edit]
set interfaces ge-0/1/0 vlan-tagging
set interfaces ge-0/1/0 encapsulation flexible-ethernet-services
set interfaces ge-0/1/0 unit 100 encapsulation vlan-vpls
set interfaces ge-0/1/0 unit 100 vlan-id 100
set interfaces ge-0/1/0 unit 100 input-vlan-map pop
set interfaces ge-0/1/0 unit 100 output-vlan-map push
set interfaces ge-0/1/0 unit 100 family vpls
set interfaces ge-0/1/0 unit 200 encapsulation vlan-vpls
set interfaces ge-0/1/0 unit 200 vlan-id 200
set interfaces ge-0/1/0 unit 200 family vpls
set interfaces ge-0/1/1 unit 0 description "PE1 to PE2"
set interfaces ge-0/1/1 unit 0 family inet address 192.0.2.4/24
set interfaces ge-0/1/1 unit 0 family iso
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces ge-0/3/0 unit 0 description "PE1 to RR"
set interfaces ge-0/3/0 unit 0 family inet address 192.0.2.7/24
set interfaces ge-0/3/0 unit 0 family iso
set interfaces ge-0/3/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.8/24
set routing-options router-id 192.0.2.8
set routing-options autonomous-system 100
set protocols mpls interface lo0.0
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 192.0.2.8
set protocols bgp group int family l2vpn auto-discovery-only
set protocols bgp group int neighbor 192.0.2.9
set protocols isis level 1 disable
set protocols isis interface all
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances vpls100 instance-type vpls
set routing-instances vpls100 interface ge-0/1/0.100
set routing-instances vpls100 route-distinguisher 192.0.2.8:100
set routing-instances vpls100 l2vpn-id l2vpn-id:100:100
set routing-instances vpls100 vrf-target target:100:100
set routing-instances vpls100 protocols vpls no-tunnel-services
set routing-instances vpls200 instance-type vpls
set routing-instances vpls200 interface ge-0/1/0.200
set routing-instances vpls200 route-distinguisher 192.0.2.8:200
set routing-instances vpls200 l2vpn-id l2vpn-id:100:200
set routing-instances vpls200 vrf-target target:100:208
set routing-instances vpls200 protocols vpls no-tunnel-services
```

On Device CE1:

```
[edit]
set interfaces ge-1/2/1 vlan-tagging
set interfaces ge-1/2/1 mtu 1400
set interfaces ge-1/2/1 unit 100 vlan-id 100
set interfaces ge-1/2/1 unit 100 family inet address 203.0.113.3/24
set interfaces ge-1/2/1 unit 200 vlan-id 200
set interfaces ge-1/2/1 unit 200 family inet address 203.0.113.2/24
set protocols ospf area 0.0.0.0 interface ge-1/2/1.100
set protocols ospf area 0.0.0.0 interface ge-1/2/1.200
```

On Router PE2:

```
[edit]
set interfaces ge-1/1/0 vlan-tagging
set interfaces ge-1/1/0 encapsulation flexible-ethernet-services
set interfaces ge-1/1/0 unit 100 encapsulation vlan-vpls
set interfaces ge-1/1/0 unit 100 vlan-id 100
set interfaces ge-1/1/0 unit 100 input-vlan-map pop
set interfaces ge-1/1/0 unit 100 output-vlan-map push
set interfaces ge-1/1/0 unit 100 family vpls
set interfaces ge-1/1/0 unit 200 encapsulation vlan-vpls
set interfaces ge-1/1/0 unit 200 vlan-id 200
set interfaces ge-1/1/0 unit 200 family vpls
set interfaces ge-1/2/1 unit 0 description "PE2 to PE1"
set interfaces ge-1/2/1 unit 0 family inet address 192.0.2.14/24
set interfaces ge-1/2/1 unit 0 family iso
set interfaces ge-1/2/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.10/24
set routing-options router-id 192.0.2.10
set routing-options autonomous-system 100
set protocols mpls interface lo0.0
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 192.0.2.10
set protocols bgp group int family l2vpn auto-discovery-only
set protocols bgp group int neighbor 192.0.2.9
set protocols isis level 1 disable
set protocols isis interface ge-1/2/1.0
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances vpls100 instance-type vpls
set routing-instances vpls100 interface ge-1/1/0.100
set routing-instances vpls100 route-distinguisher 192.0.2.10:100
set routing-instances vpls100 l2vpn-id l2vpn-id:100:100
set routing-instances vpls100 vrf-target target:100:100
set routing-instances vpls100 protocols vpls no-tunnel-services
set routing-instances vpls200 instance-type vpls
set routing-instances vpls200 interface ge-1/1/0.200
set routing-instances vpls200 route-distinguisher 192.0.2.10:200
set routing-instances vpls200 l2vpn-id l2vpn-id:100:200
set routing-instances vpls200 vrf-target target:100:208
set routing-instances vpls200 protocols vpls no-tunnel-services
```

On Device CE2:

```
[edit]
set interfaces ge-1/1/0 vlan-tagging
set interfaces ge-1/1/0 mtu 1400
set interfaces ge-1/1/0 unit 100 vlan-id 100
set interfaces ge-1/1/0 unit 100 family inet address 203.0.113.15/24
set interfaces ge-1/1/0 unit 200 vlan-id 200
set interfaces ge-1/1/0 unit 200 family inet address 203.0.113.16/24
set protocols ospf area 0.0.0.0 interface ge-1/1/0.100
set protocols ospf area 0.0.0.0 interface ge-1/1/0.200
```

On Router RR:

```
[edit]
set interfaces ge-1/3/2 unit 0 description "RR to PE1"
set interfaces ge-1/3/2 unit 0 family inet address 192.0.2.17/24
set interfaces ge-1/3/2 unit 0 family iso
set interfaces ge-1/3/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.9/24
set routing-options router-id 192.0.2.9
set routing-options autonomous-system 100
set protocols bgp group int type internal
set protocols bgp group int local-address 192.0.2.9
set protocols bgp group int family l2vpn auto-discovery-only
set protocols bgp group int cluster 198.51.100.0
set protocols bgp group int neighbor 192.0.2.8
set protocols bgp group int neighbor 192.0.2.10
set protocols isis level 1 disable
set protocols isis interface all
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
```

Router PE1

Step-by-Step Procedure

To configure Router PE1:

1. Configure the interfaces, the interface encapsulation, and the protocol families.

```
[edit]
user@PE1# edit interfaces
[edit interfaces]
user@PE1# set ge-0/1/0 encapsulation flexible-ethernet-services
user@PE1# set ge-0/1/0 unit 100 encapsulation vlan-vpls
user@PE1# set ge-0/1/0 unit 100 family vpls
user@PE1# set ge-0/1/0 unit 200 encapsulation vlan-vpls
user@PE1# set ge-0/1/0 unit 200 family vpls
user@PE1# set ge-0/1/1 unit 0 description "PE1 to PE2"
user@PE1# set ge-0/1/1 unit 0 family inet address 192.0.2.4/24
user@PE1# set ge-0/1/1 unit 0 family iso
user@PE1# set ge-0/1/1 unit 0 family mpls
user@PE1# set ge-0/3/0 unit 0 description "PE1 to RR"
user@PE1# set ge-0/3/0 unit 0 family inet address 192.0.2.7/24
user@PE1# set ge-0/3/0 unit 0 family iso
user@PE1# set ge-0/3/0 unit 0 family mpls
user@PE1# set lo0 unit 0 family inet address 192.0.2.8/24
```

2. Configure the VLANs.

```
[edit interfaces]
user@PE1# set ge-0/1/0 vlan-tagging
user@PE1# set ge-0/1/0 unit 100 vlan-id 100
user@PE1# set ge-0/1/0 unit 100 input-vlan-map pop
user@PE1# set ge-0/1/0 unit 100 output-vlan-map push
user@PE1# set ge-0/1/0 unit 200 vlan-id 200
user@PE1# exit
```

3. Configure the protocol-independent properties.

   We recommend that the router ID be the same as the local address. (See the `local-address` statement in Step "4" on page 921.)

```
[edit]
user@PE1# edit routing-options
[edit routing-options]
user@PE1# set router-id 192.0.2.8
user@PE1# set autonomous-system 100
user@PE1# exit
```

4. Configure IBGP, including the `auto-discovery-only` statement.

```
[edit]
user@PE1# edit protocols
[edit protocols]
user@PE1# set bgp group int type internal
user@PE1# set bgp group int local-address 192.0.2.8
user@PE1# set bgp group int family l2vpn auto-discovery-only
user@PE1# set bgp group int neighbor 192.0.2.9
```

5. Configure MPLS, LDP, and an IGP.

```
[edit protocols]
user@PE1# set mpls interface lo0.0
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable
user@PE1# set isis level 1 disable
```

```
user@PE1# set isis interface all
user@PE1# set isis interface fxp0.0 disable
user@PE1# set isis interface lo0.0
user@PE1# set ldp interface all
user@PE1# set ldp interface fxp0.0 disable
user@PE1# set ldp interface lo0.0
user@PE1# exit
```

6. Configure the routing instances.

   The `no-tunnel-services` statement is required if you are using LSI interfaces for VPLS instead of `vt` interfaces.

```
[edit]
user@PE1# edit routing-instances
[edit routing-instances]
user@PE1# set vpls100 instance-type vpls
user@PE1# set vpls100 interface ge-0/1/0.100
user@PE1# set vpls100 route-distinguisher 192.0.2.8:100
user@PE1# set vpls100 l2vpn-id l2vpn-id:100:100
user@PE1# set vpls100 vrf-target target:100:100
user@PE1# set vpls100 protocols vpls no-tunnel-services
user@PE1# set vpls200 instance-type vpls
user@PE1# set vpls200 interface ge-0/1/0.200
user@PE1# set vpls200 route-distinguisher 192.0.2.8:200
user@PE1# set vpls200 l2vpn-id l2vpn-id:100:200
user@PE1# set vpls200 vrf-target target:100:208
user@PE1# set vpls200 protocols vpls no-tunnel-services
```

7. 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 interfaces`, `show protocols`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE1# show interfaces
ge-0/1/0 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 100 {
        encapsulation vlan-vpls;
        vlan-id 100;
        input-vlan-map pop;
        output-vlan-map push;
        family vpls;
    }
    unit 200 {
        encapsulation vlan-vpls;
        vlan-id 200;
        family vpls;
    }
}
ge-0/1/1 {
    unit 0 {
        description "PE1 to PE2";
        family inet {
            address 192.0.2.4/24;
        }
        family iso;
        family mpls;
    }
}
ge-0/3/0 {
    unit 0 {
        description "PE1 to RR";
        family inet {
            address 192.0.2.7/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.8/24;
        }
    }
}
```

```
user@PE1# **show protocols**
mpls {
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
bgp {
    group int {
        type internal;
        local-address 192.0.2.8;
        family l2vpn {
            auto-discovery-only;
        }
        neighbor 192.0.2.9;
    }
}
isis {
    level 1 disable;
    interface all;
    interface lo0.0;
    interface fxp0 disable;
}
ldp {
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
```

```
user@PE1# show routing-options
router-id 192.0.2.8;
autonomous-system 100;
```

```
user@PE1# show routing-instances
vpls100 {
    instance-type vpls;
    interface ge-0/1/0.100;
    route-distinguisher 192.0.2.8:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        vpls {
            no-tunnel-services;
        }
    }
}
vpls200 {
    instance-type vpls;
    interface ge-0/1/0.200;
    route-distinguisher 192.0.2.8:200;
    l2vpn-id l2vpn-id:100:200;
    vrf-target target:100:208;
    protocols {
        vpls {
            no-tunnel-services;
        }
    }
}
```

Device CE1

Step-by-Step Procedure

To configure Device CE1:

1. Configure interface addresses and the interface maximum transmission unit (MTU).

```
[edit]
user@CE1# edit interfaces
[edit interfaces]
user@CE1# set ge-1/2/1 mtu 1400
user@CE1# set ge-1/2/1 unit 100 family inet address 203.0.113.3/24
user@CE1# set ge-1/2/1 unit 200 family inet address 203.0.113.2/24
```

2. Configure VLANs.

```
[edit interfaces]
user@CE1# set ge-1/2/1 vlan-tagging
user@CE1# set ge-1/2/1 unit 100 vlan-id 100
user@CE1# set ge-1/2/1 unit 200 vlan-id 200
user@CE1# exit
```

3. Configure an IGP.

```
user@CE1# edit protocols
[edit protocols]
user@CE1# set ospf area 0.0.0.0 interface ge-1/2/1.100
user@CE1# set ospf area 0.0.0.0 interface ge-1/2/1.200
user@CE1# exit
```

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

```
[edit]
user@CE1# commit
```

Results

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

```
user@CE1# show interfaces
ge-1/2/1 {
    vlan-tagging;
    mtu 1400;
    unit 100 {
        vlan-id 100;
        family inet {
            address 203.0.113.3/24;
        }
    }
    unit 200 {
        vlan-id 200;
        family inet {
            address 203.0.113.2/24;
        }
    }
}
```

```
user@CE1# show protocols
ospf {
    area 0.0.0.0 {
        interface ge-1/2/1.100;
        interface ge-1/2/1.200;
    }
}
```

Router PE2

Step-by-Step Procedure

To configure Router PE2:

1. Configure the interfaces, the interface encapsulation, and the protocol families.

```
[edit]
user@PE2# edit interfaces
[edit interfaces]
user@PE2# set ge-1/1/0 encapsulation flexible-ethernet-services
user@PE2# set ge-1/1/0 unit 100 encapsulation vlan-vpls
user@PE2# set ge-1/1/0 unit 100 family vpls
user@PE2# set ge-1/1/0 unit 200 encapsulation vlan-vpls
user@PE2# set ge-1/1/0 unit 200 family vpls
```

```
user@PE2# set ge-1/2/1 unit 0 description "PE2 to PE1"
user@PE2# set ge-1/2/1 unit 0 family inet address 192.0.2.14/24
user@PE2# set ge-1/2/1 unit 0 family iso
user@PE2# set ge-1/2/1 unit 0 family mpls
user@PE2# set lo0 unit 0 family inet address 192.0.2.10/24
```

2. Configure the VLANs.

```
[edit interfaces]
user@PE2# set ge-1/1/0 vlan-tagging
user@PE2# set ge-1/1/0 unit 100 vlan-id 100
user@PE2# set ge-1/1/0 unit 100 input-vlan-map pop
user@PE2# set ge-1/1/0 unit 100 output-vlan-map push
user@PE2# set ge-1/1/0 unit 200 vlan-id 200
user@PE2# exit
```

3. Configure the protocols-independent properties.

   We recommend that the router ID be the same as the local address. (See the `local-address` statement in Step "4" on page 928.)

```
[edit]
user@PE2# edit routing-options
[edit routing-options]
user@PE2# set router-id 192.0.2.10
user@PE2# set autonomous-system 100
```

4. Configure IBGP, including the `auto-discovery-only` statement.

```
[edit]
user@PE2# edit protocols
[edit protocols]
user@PE2# set bgp group int type internal
user@PE2# set bgp group int local-address 192.0.2.10
user@PE2# set bgp group int family l2vpn auto-discovery-only
user@PE2# set bgp group int neighbor 192.0.2.9
```

5. Configure MPLS, LDP, and an IGP.

```
[edit protocols]
user@PE2# set mpls interface lo0.0
user@PE2# set mpls interface all
user@PE2# set mpls interface fxp0.0 disable
user@PE2# set isis level 1 disable
user@PE2# set isis interface ge-1/2/1.0
user@PE2# set isis interface lo0.0
user@PE2# set ldp interface all
user@PE2# set ldp interface fxp0.0 disable
user@PE2# set ldp interface lo0.0
user@PE2# exit
```

6. Configure the routing instances.

   The `no-tunnel-services` statement is required if you are using LSI interfaces for VPLS instead of `vt` interfaces.

```
[edit]
user@PE2# edit routing-instances
[edit routing-instances]
user@PE2# set vpls100 instance-type vpls
user@PE2# set vpls100 interface ge-1/1/0.100
user@PE2# set vpls100 route-distinguisher 192.0.2.10:100
user@PE2# set vpls100 l2vpn-id l2vpn-id:100:100
user@PE2# set vpls100 vrf-target target:100:100
user@PE2# set vpls100 protocols vpls no-tunnel-services
user@PE2# set vpls200 instance-type vpls
user@PE2# set vpls200 interface ge-1/1/0.200
user@PE2# set vpls200 route-distinguisher 192.0.2.10:200
user@PE2# set vpls200 l2vpn-id l2vpn-id:100:200
user@PE2# set vpls200 vrf-target target:100:208
user@PE2# set vpls200 protocols vpls no-tunnel-services
```

7. 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 interfaces`, `show protocols`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE2# show interfaces
ge-1/1/0 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 100 {
        encapsulation vlan-vpls;
        vlan-id 100;
        input-vlan-map pop;
        output-vlan-map push;
        family vpls;
    }
    unit 200 {
        encapsulation vlan-vpls;
        vlan-id 200;
        family vpls;
    }
}
ge-1/2/1 {
    unit 0 {
        description "PE2 to PE1";
        family inet {
            address 192.0.2.14/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.10/24;
        }
    }
}
```

```
user@PE2# show protocols
mpls {
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
bgp {
    group int {
        type internal;
        local-address 192.0.2.10;
        family l2vpn {
            auto-discovery-only;
        }
        neighbor 192.0.2.9;
    }
}
isis {
    level 1 disable;
    interface ge-1/2/1.0;
    interface lo0.0;
}
ldp {
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
```

```
user@PE2# show routing-options
router-id 192.0.2.10;
autonomous-system 100;
```

```
user@PE2# show routing-instances
vpls100 {
    instance-type vpls;
    interface ge-1/1/0.100;
    route-distinguisher 192.0.2.10:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        vpls {
            no-tunnel-services;
        }
    }
}
vpls200 {
    instance-type vpls;
    interface ge-1/1/0.200;
    route-distinguisher 192.0.2.10:200;
    l2vpn-id l2vpn-id:100:200;
    vrf-target target:100:208;
    protocols {
        vpls {
            no-tunnel-services;
        }
    }
}
```

Device CE2

Step-by-Step Procedure

To configure Device CE2:

1. Configure VLAN interfaces.

```
[edit]
user@CE2# edit interfaces ge-1/1/0
[edit interfaces ge-1/1/0]
user@CE2# set vlan-tagging
user@CE2# set mtu 1400
user@CE2# set unit 100 vlan-id 100
user@CE2# set unit 100 family inet address 203.0.113.15/24
user@CE2# set unit 200 vlan-id 200
user@CE2# set unit 200 family inet address 203.0.113.16/24
user@CE2# exit
```

2. Configure OSPF on the interfaces.

```
[edit]
user@CE2# edit protocols ospf area 0.0.0.0
[edit protocols ospf area 0.0.0.0]
user@CE2# set interface ge-1/1/0.100
user@CE2# set interface ge-1/1/0.200
user@CE2# exit
```

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

```
[edit]
user@CE2# commit
```

Results

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

```
user@CE2# show interfaces
ge-1/1/0 {
    vlan-tagging;
    mtu 1400;
    unit 100 {
        vlan-id 100;
        family inet {
            address 203.0.113.15/24;
        }
    }
    unit 200 {
        vlan-id 200;
        family inet {
            address 203.0.113.16/24;
        }
    }
}
```

```
user@CE2# show protocols
ospf {
    area 0.0.0.0 {
        interface ge-1/1/0.100;
        interface ge-1/1/0.200;
    }
}
```

Router RR

Step-by-Step Procedure

To configure Router RR:

1. Configure interface addresses and the protocol families.

   ```
   [edit]
   user@RR# edit interfaces
   [edit interfaces]
   user@RR# set ge-1/3/2 unit 0 description "RR to PE1"
   user@RR# set ge-1/3/2 unit 0 family inet address 192.0.2.17/24
   user@RR# set ge-1/3/2 unit 0 family iso
   user@RR# set ge-1/3/2 unit 0 family mpls
   user@RR# set lo0 unit 0 family inet address 192.0.2.9/24
   user@RR# exit
   ```

2. Configure the autonomous systems and the router ID.

   ```
   [edit]
   user@RR# edit routing-options
   [edit routing-options]
   user@RR# set autonomous-system 100
   user@RR# set router-id 192.0.2.9
   user@RR# exit
   ```

3. Configure BGP and set this router to be the route reflector. Route reflection is optional for FEC 129.

```
[edit]
user@RR# edit protocols bgp group int
[edit protocols bgp group int]
user@RR# set type internal
user@RR# set local-address 192.0.2.9
user@RR# set family l2vpn auto-discovery-only
user@RR# set cluster 198.51.100.0
user@RR# set neighbor 192.0.2.8
user@RR# set neighbor 192.0.2.10
user@RR# exit
```

4. Configure IS-IS for the IGP.

```
[edit]
user@RR# edit protocols isis
[edit protocols isis]
user@RR# set level 1 disable
user@RR# set interface all
user@RR# set interface fxp0.0 disable
user@RR# set interface lo0.0
user@RR# exit
```

5. Configure LDP for the MPLS signaling protocol.

```
[edit]
user@RR# edit protocols ldp
[edit protocols ldp]
user@RR# set interface all
user@RR# set interface fxp0.0 disable
user@RR# set interface lo0.0
user@RR# exit
```

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

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

Results

From configuration mode, confirm your configuration by entering the `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@RR# show interfaces
ge-1/3/2 {
    unit 0 {
        description "RR to PE1";
        family inet {
            address 192.0.2.17/24;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.9/24;;
        }
    }
}
```

```
user@RR# show protocols
bgp {
    group int {
        type internal;
        local-address 192.0.2.9;
        family l2vpn {
            auto-discovery-only;
        }
        cluster 198.51.100.0;
        neighbor 192.0.2.8;
        neighbor 192.0.2.10;
    }
}
isis {
    level 1 disable;
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
ldp {
    interface lo0.0;
    interface all;
    interface fxp0 disable;
}
```

```
user@RR# show routing-options
router-id 192.0.2.9;
autonomous-system 100;
```

Verification

To verify the operation, use the following commands:

- `show route extensive`
- `show route advertising-protocol bgp` *`neighbor`*
- `show route receive-protocol bgp` *`neighbor`*
- `show route table bgp.l2vpn.0`
- `show route table vpls100.l2vpn.0`
- `show route table vpls200.l2vpn.0`
- `show vpls connections extensive`
- `show vpls mac-table detail`
- `show vpls statistics`

`AD` in the routing table output indicates autodiscovery NLRI.

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | **938**

Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS | **632**

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups

IN THIS SECTION

- Requirements | 938
- Overview | 938
- Configuration | 939
- Verification | 947

This example describes how to configure user-defined mesh groups for BGP autodiscovery for LDP VPLS, as specified in forwarding equivalency class (FEC) 129. FEC 129 uses BGP autodiscovery to convey endpoint information, so you do not need to manually configure pseudowires. You configure mesh groups on the border router to group the sets of PE routers that are automatically fully meshed and that share the same signaling protocol, either BGP or LDP. You can configure multiple mesh groups to map each fully meshed LDP-signaled or BGP-signaled VPLS domain to a mesh group.

Requirements

Before you begin, configure BGP autodiscovery for LDP VPLS. See "Example: Configuring BGP Autodiscovery for LDP VPLS" on page 913.

The hardware and software requirements for this example are the same as the requirements for the *Example: Configuring BGP Autodiscovery for LDP VPLS*. You will need to adapt the example configuration to the topology used in this example.

Overview

IN THIS SECTION

- Topology Diagram | 939

Configuration for a mesh group for FEC 129 is very similiar to the mesh-group configuration for FEC 128.

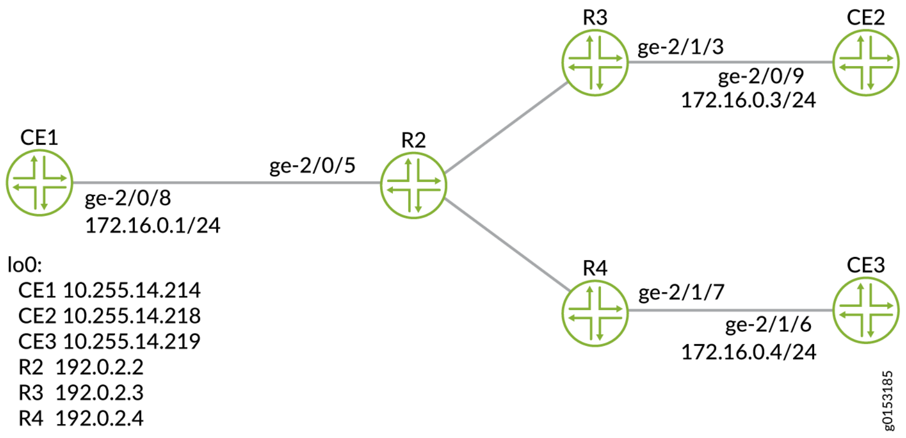
Note the following differences for FEC 129:

- Each user-defined mesh group must have a unique route distinguisher. Do not use the route distinguisher that is defined for the default mesh group at the `[edit routing-instances]` hierarchy level.
- Each user-defined mesh group must have its own import and export route target.
- Each user-defined mesh group can have a unique Layer 2 VPN ID. By default, all the mesh groups that are configured for a VPLS routing instance use the same Layer 2 VPN ID as the one that you configure at the `[edit routing-instances]` hierarchy level.

Topology Diagram

Figure 68 on page 939 shows a topology that includes a user-defined mesh group.

Figure 68: BGP Autodiscovery for LDP VPLS with a User-Defined Mesh Group



Configuration

IN THIS SECTION

- CLI Quick Configuration | 939
- Procedure | 942

CLI Quick Configuration

To quickly configure a mesh group, copy the following commands, remove any line breaks, and then paste the commands into the CLI of each device.

Device CE1

```
set interfaces ge-2/0/8 unit 0 family inet address 172.16.0.1/24
set interfaces lo0 unit 0 family inet address 10.255.14.214/32
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 CE3

```
set interfaces ge-2/0/9 unit 0 family inet address 172.16.0.3/24
set interfaces lo0 unit 0 family inet address 10.255.14.218/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/9.0
```

Device CE4

```
set interfaces ge-2/1/6 unit 0 family inet address 172.16.0.4/24
set interfaces lo0 unit 0 family inet address 10.255.14.219/32
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/6.0
```

Device R2

```
set interfaces ge-2/0/5 encapsulation ethernet-vpls
set interfaces ge-2/0/5 unit 0 description to_CE1
set interfaces ge-2/0/5 unit 0 family vpls
set interfaces ge-2/0/10 unit 0 description to_R3
set interfaces ge-2/0/10 unit 0 family inet address 10.10.4.2/30
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/0/11 unit 0 description to_R4
set interfaces ge-2/0/11 unit 0 family inet address 10.10.5.1/30
set interfaces ge-2/0/11 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.2/24
set protocols mpls interface ge-2/0/10.0
set protocols mpls interface ge-2/0/11.0
set protocols bgp local-address 192.0.2.2
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe connect-retry-interval 1
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.3
set protocols bgp group pe-pe neighbor 192.0.2.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-2/0/10.0
set protocols ospf area 0.0.0.0 interface ge-2/0/11.0
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface ge-2/0/11.0
set protocols ldp interface lo0.0
set routing-instances inst512 instance-type vpls
set routing-instances inst512 interface ge-2/0/5.0
set routing-instances inst512 route-distinguisher 100:100
set routing-instances inst512 l2vpn-id l2vpn-id:1:2
set routing-instances inst512 vrf-target target:1:1
set routing-instances inst512 protocols vpls mesh-group metro1 vrf-target target:2:1
set routing-instances inst512 protocols vpls mesh-group metro1 route-distinguisher 100:200
set routing-options autonomous-system 64510
```

Device R3

```
set interfaces ge-2/0/10 unit 0 description to_R2
set interfaces ge-2/0/10 unit 0 family inet address 10.10.4.1/30
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/1/3 encapsulation ethernet-vpls
set interfaces ge-2/1/3 unit 0 description to_CE2
set interfaces ge-2/1/3 unit 0 family vpls
set interfaces lo0 unit 0 family inet address 192.0.2.3/24
set protocols mpls interface ge-2/0/10.0
set protocols bgp local-address 192.0.2.3
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe connect-retry-interval 1
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols bgp group pe-pe neighbor 192.0.2.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/0/10.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances inst512 instance-type vpls
set routing-instances inst512 interface ge-2/1/3.0
set routing-instances inst512 route-distinguisher 100:100
set routing-instances inst512 l2vpn-id l2vpn-id:1:2
set routing-instances inst512 vrf-target target:1:1
set routing-instances inst512 protocols vpls
set routing-options autonomous-system 64510
```

Device R4

```
set interfaces ge-2/0/10 unit 0 description to_R2
set interfaces ge-2/0/10 unit 0 family inet address 10.10.5.2/30
set interfaces ge-2/0/10 unit 0 family mpls
set interfaces ge-2/1/7 encapsulation ethernet-vpls
set interfaces ge-2/1/7 unit 0 description to_CE3
set interfaces ge-2/1/7 unit 0 family vpls
set interfaces lo0 unit 0 family inet address 192.0.2.4/24
set protocols mpls interface ge-2/0/10.0
set protocols bgp local-address 192.0.2.4
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe connect-retry-interval 1
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols bgp group pe-pe neighbor 192.0.2.3
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-2/0/10.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface ge-2/0/10.0
set protocols ldp interface lo0.0
set routing-instances inst512 instance-type vpls
set routing-instances inst512 interface ge-2/1/7.0
set routing-instances inst512 route-distinguisher 100:100
set routing-instances inst512 l2vpn-id l2vpn-id:1:2
set routing-instances inst512 vrf-target target:1:1
set routing-instances inst512 protocols vpls
set routing-options autonomous-system 64510
```

Procedure

Step-by-Step Procedure

To configure a mesh group:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set ge-2/0/5 encapsulation ethernet-vpls
user@R2# set ge-2/0/5 unit 0 description to_CE1
user@R2# set ge-2/0/5 unit 0 family vpls
user@R2# set ge-2/0/10 unit 0 description to_R3
user@R2# set ge-2/0/10 unit 0 family inet address 10.10.4.2/30
user@R2# set ge-2/0/10 unit 0 family mpls
user@R2# set ge-2/0/11 unit 0 description to_R4
user@R2# set ge-2/0/11 unit 0 family inet address 10.10.5.1/30
user@R2# set ge-2/0/11 unit 0 family mpls
user@R2# set lo0 unit 0 family inet address 192.0.2.2/24
```

2. Configure MPLS on the interfaces.

```
[edit protocols mpls]
user@R2# set interface ge-2/0/10.0
user@R2# set interface ge-2/0/11.0
```

3. Configure BGP.

```
[edit protocols bgp]
user@R2# set local-address 192.0.2.2
[edit protocols bgp group pe-pe]
user@R2# set type internal
user@R2# set connect-retry-interval 1
user@R2# set family l2vpn auto-discovery-only
user@R2# set family l2vpn signaling
user@R2# set neighbor 192.0.2.3
user@R2# set neighbor 192.0.2.4
```

4. Set the import and export route target for the default mesh group.

```
[edit protocols ospf]
user@R2# set traffic-engineering
user@R2# set area 0.0.0.0 interface lo0.0 passive
```

```
user@R2# set area 0.0.0.0 interface ge-2/0/10.0
user@R2# set area 0.0.0.0 interface ge-2/0/11.0
```

5. Configure LDP on the core-facing interfaces and on the loopback interface.

```
[edit protocols ldp]
user@R2# set interface ge-2/0/10.0
user@R2# set interface ge-2/0/11.0
user@R2# set interface lo0.0
```

6. Configure the VPLS routing instance.

   Make sure that the route distinguisher in the mesh group is unique.

```
[edit routing-instances inst512]
user@R2# set instance-type vpls
user@R2# set interface ge-2/0/5.0
user@R2# set route-distinguisher 100:100
user@R2# set l2vpn-id l2vpn-id:1:2
user@R2# set vrf-target target:1:1
user@R2# set protocols vpls mesh-group metro1 vrf-target target:2:1
user@R2# set protocols vpls mesh-group metro1 route-distinguisher 100:200
```

7. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@R2# set autonomous-system 64510
```

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

```
[edit]
user@R2# commit
```

Results

From configuration mode, confirm your configuration by entering the `show routing-instances` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
ge-2/0/5 {
    encapsulation ethernet-vpls;
    unit 0 {
        description PE1_to_CE1;
        family vpls;
    }
}
ge-2/0/10 {
    unit 0{
        description to_R3;
        family inet {
            address 10.10.4.2/30;
        }
        family mpls;
    }
}
ge-2/0/11 {
    unit 0 {
        description to_R4;
        family inet {
            address 10.10.5.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0{
        family inet {
            address 192.0.2.2/24;
        }
    }
}
```

```
user@R2# show protocols
mpls {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
}
bgp {
    local-address 192.0.2.2;
    group pe-pe {
        type internal;
        connect-retry-interval 1;
        family l2vpn {
            auto-discovery-only;
            signaling;
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.4;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ge-2/0/10.0;
        interface ge-2/0/11.0;
    }
}
ldp {
    interface ge-2/0/10.0;
    interface ge-2/0/11.0;
    interface lo0.0;
}
```

```
user@R2# show routing-instances
inst512 {
    instance-type vpls;
    interface ge-2/0/5.0;
    route-distinguisher 100:100;
    l2vpn-id l2vpn-id:1:2;
    vrf-target target:1:1;
    protocols {
        vpls {
            mesh-group metro1 {
                vrf-target target:2:1;
                route-distinguisher 100:200;
            }
        }
    }
}
```

```
user@R2# show routing-options
autonomous-system 64510;
```

Verification

IN THIS SECTION

- Verifying the Routes | 947
- Checking Connectivity | 950
- Checking the VPLS Connections | 951
- Display Learned VPLS MAC Address Information | 952

Confirm that the configuration is working properly.

Verifying the Routes

Purpose

Verify that the expected routes are learned.

Action

From operational mode, enter the `show route` command.

```
user@R2> show route
inet.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2/24         *[Direct/0] 4d 02:42:47
                    > via lo0.0
192.0.2.3/24         *[OSPF/10] 4d 02:41:56, metric 1
                    > to 10.10.4.1 via ge-2/0/10.0
192.0.2.4/24         *[OSPF/10] 4d 02:42:01, metric 1
                    > to 10.10.5.2 via ge-2/0/11.0
10.10.3.2/24       *[Local/0] 4d 02:42:47
                      Reject
10.10.4.0/30       *[Direct/0] 4d 02:42:46
                    > via ge-2/0/10.0
10.10.4.2/32       *[Local/0] 4d 02:42:47
                      Local via ge-2/0/10.0
10.10.5.0/30       *[Direct/0] 4d 02:42:46
                    > via ge-2/0/11.0
10.10.5.1/32       *[Local/0] 4d 02:42:47
                      Local via ge-2/0/11.0
203.0.113.0/24       *[OSPF/10] 4d 02:42:49, metric 1
                      MultiRecv

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3/24         *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.4.1 via ge-2/0/10.0
192.0.2.4/24         *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.5.2 via ge-2/0/11.0

mpls.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 4d 02:42:49, metric 1
                      Receive
1                  *[MPLS/0] 4d 02:42:49, metric 1
                      Receive
```

```
2                  *[MPLS/0] 4d 02:42:49, metric 1
                      Receive
13                 *[MPLS/0] 4d 02:42:49, metric 1
                      Receive
299776             *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.5.2 via ge-2/0/11.0, Pop
299776(S=0)        *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.5.2 via ge-2/0/11.0, Pop
299792             *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.4.1 via ge-2/0/10.0, Pop
299792(S=0)        *[LDP/9] 4d 02:01:06, metric 1
                    > to 10.10.4.1 via ge-2/0/10.0, Pop
800000             *[VPLS/7] 4d 02:01:05
                    > via vt-2/0/10.185597952, Pop
800001             *[VPLS/7] 4d 02:01:05
                    > via vt-2/0/10.185597953, Pop
vt-2/0/10.185597953*[VPLS/7] 4d 02:01:05, metric2 1
                    > to 10.10.5.2 via ge-2/0/11.0, Push 800001
vt-2/0/10.185597952*[VPLS/7] 4d 02:01:05, metric2 1
                    > to 10.10.4.1 via ge-2/0/10.0, Push 800001

bgp.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100:100:192.0.2.3/96 AD
                   *[BGP/170] 4d 02:32:41, localpref 100, from 192.0.2.3
                      AS path: I, validation-state: unverified
                    > to 10.10.4.1 via ge-2/0/10.0
100:100:192.0.2.4/96 AD
                   *[BGP/170] 4d 02:32:41, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    > to 10.10.5.2 via ge-2/0/11.0

inst512.l2vpn.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100:100:192.0.2.2/96 AD
                   *[VPLS/170] 4d 02:01:05, metric2 1
                      Indirect
100:100:192.0.2.3/96 AD
                   *[BGP/170] 4d 02:32:41, localpref 100, from 192.0.2.3
                      AS path: I, validation-state: unverified
                    > to 10.10.4.1 via ge-2/0/10.0
```

```
100:100:192.0.2.4/96 AD
                   *[BGP/170] 4d 02:32:41, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    > to 10.10.5.2 via ge-2/0/11.0
100:200:192.0.2.2/96 AD
                   *[VPLS/170] 4d 02:01:05, metric2 1
                      Indirect
192.0.2.3:NoCtrlWord:5:1:2:192.0.2.2:192.0.2.3/176
                   *[VPLS/7] 4d 02:01:05, metric2 1
                    > to 10.10.4.1 via ge-2/0/10.0
192.0.2.3:NoCtrlWord:5:1:2:192.0.2.3:192.0.2.2/176
                   *[LDP/9] 4d 02:01:05
                      Discard
192.0.2.4:NoCtrlWord:5:1:2:192.0.2.2:192.0.2.4/176
                   *[VPLS/7] 4d 02:01:05, metric2 1
                    > to 10.10.5.2 via ge-2/0/11.0
192.0.2.4:NoCtrlWord:5:1:2:192.0.2.4:192.0.2.2/176
                   *[LDP/9] 4d 02:01:05
                      Discard

ldp.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.3:NoCtrlWord:5:1:2:192.0.2.3:192.0.2.2/176
                   *[LDP/9] 4d 02:01:05
                      Discard
192.0.2.4:NoCtrlWord:5:1:2:192.0.2.4:192.0.2.2/176
                   *[LDP/9] 4d 02:01:05
                      Discard
```

Meaning

The output shows all the learned routes, including the autodiscovered (AD) routes.

Checking Connectivity

Purpose

Verify that Device CE1 can ping Device CE2 and Device CE3.

Action

```
user@CE1> ping 10.255.14.218
PING 10.255.14.218 (10.255.14.218): 56 data bytes
64 bytes from 10.255.14.218: icmp_seq=0 ttl=64 time=0.787 ms
64 bytes from 10.255.14.218: icmp_seq=1 ttl=64 time=0.651 ms
^C
--- 10.255.14.218 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.651/0.719/0.787/0.068 ms
```

```
user@CE1> ping 10.255.14.219
PING 10.255.14.219 (10.255.14.219): 56 data bytes
64 bytes from 10.255.14.219: icmp_seq=0 ttl=64 time=1.054 ms
64 bytes from 10.255.14.219: icmp_seq=1 ttl=64 time=0.669 ms
^C
--- 10.255.14.219 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.669/0.862/1.054/0.193 ms
```

Meaning

The output shows that VPLS is operational.

Checking the VPLS Connections

Purpose

Make sure that all of the FEC 129 VPLS connections come up correctly.

Action

```
user@R2> show vpls connections
Instance: inst512
  L2vpn-id: 1:2
  Local-id: 192.0.2.2
  Mesh-group connections: __ves__
    Remote-id                 Type  St     Time last up          # Up trans
    192.0.2.4                   rmt   Up     Oct 26 15:11:56 2012           1
```

```
        Remote PE: 192.0.2.4, Negotiated control-word: No
        Incoming label: 800001, Outgoing label: 800001
        Local interface: vt-2/0/10.185597953, Status: Up, Encapsulation: ETHERNET
          Description: Intf - vpls inst512 local-id 192.0.2.2 remote-id 192.0.2.4 neighbor 
192.0.2.4
    192.0.2.3                   rmt   Up     Oct 26 15:11:56 2012           1
        Remote PE: 192.0.2.3, Negotiated control-word: No
        Incoming label: 800000, Outgoing label: 800001
        Local interface: vt-2/0/10.185597952, Status: Up, Encapsulation: ETHERNET
          Description: Intf - vpls inst512 local-id 192.0.2.2 remote-id 192.0.2.3 neighbor 
192.0.2.3
```

Meaning

As expected, the connections are up.

Display Learned VPLS MAC Address Information

Purpose

Verify that all CE devices' MAC addresses are learned and installed.

Action

```
user@R2> show vpls mac-table

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Logical system   : R2
Routing instance : inst512
 Bridging domain : __inst512__, VLAN : NA
   MAC                 MAC      Logical          NH     RTR
   address             flags    interface        Index  ID
   00:21:59:0f:35:32   D        ge-2/0/5.0
   00:21:59:0f:35:33   D        vt-2/0/10.185597952
   00:21:59:0f:35:d5   D        vt-2/0/10.185597953
```

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | 913

Configuring Interoperability Between BGP Signaling and LDP Signaling in VPLS | 632

VPLS Multihoming Reactions to Network Failures

VPLS multihoming is designed to protect customer sites from a loss of network connectivity in the event of the following types of network failures:

- Link failure between the CE device and the PE router—BGP on the PE router is notified when the link goes down. BGP sets the circuit status vector bit in the MP_REACH_NLRI to indicate that the circuit is down.

  If all of the VPLS local attachment circuits are down, then BGP modifies the down bit in the VPLS advertisement Layer2-Extended-Community to indicate that the customer site is down. When the bit is modified, BGP advertises the route to all of the remote PE routers to notify them that the circuit (and site) is down. Each of the remote PE routers run the BGP and VPLS path selection procedures again and reroute the VPLS pseudowires as needed.

- MPLS connectivity failure to the remote PE router—On the multihomed PE router, BGP discovers that MPLS cannot connect to the BGP next hop in the service provider's network. BGP modifies the circuit status vector bit in the MP_REACH_NLRI to indicate that the LSP is down. Once the bit is modified, BGP readvertises the route to all of the remote PE routers to notify them that connectivity from the local site to the remote site is down.

  The remote PE routers each run the BGP and VPLS path selection procedures again. With the LSP to the original multihomed PE router down, the remote PE routers designate the backup multihomed PE router as the VE device for the multihomed customer site. The pseudowires to and from the remote PE routers are then rerouted to the backup multihomed PE router.

- PE router failure—When either the multihomed PE router or the BGP process running on it fails, the remote PE routers detect the expiration of the holdtimer, bring down their peering sessions, and delete the Layer 2 advertisements from that multihomed PE router. The remote PE routers each run the BGP and VPLS path selection procedures again and reroute their pseudowires to the backup multihomed PE router.

  Alternatively, the remote PE routers could discover that the BGP next hop, represented by the failed multihomed PE router, is unreachable. For this case, the remote PE routers mark the Layer 2 routes advertised by the multihomed PE router as unreachable. The remote PE routers each run the BGP and VPLS path selection procedures again and reroute their pseudowires to the backup multihomed PE router.

The remote PE routers behave in the same manner if you reconfigure the local preference attribute of the primary multihomed PE router (effectively performing an administrative failover to the backup multihomed PE router). On the primary multihomed PE router, BGP advertises a Layer 2 update with the new local preference attribute to all of the remote PE routers. The remote PE routers each run the BGP and VPLS path selection procedures again and reroute their pseudowires to the backup multihomed PE router.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Configuring VPLS Multihoming

IN THIS SECTION

- VPLS Multihomed Site Configuration | **955**
- VPLS Single-Homed Site Configuration | **957**

VPLS multihoming allows you to connect a customer site to multiple PE routers to provide redundant connectivity while preventing the formation of Layer 2 loops in the service provider's network. A VPLS site multihomed to two or more PE routers provides redundant connectivity in the event of a PE router-to-CE device link failure or the failure of a PE router. For more information about VPLS multihoming, see "VPLS Multihoming Overview" on page 885.

NOTE: If you want to enable multihoming for a VPLS routing instance, you cannot also enable LDP signaling. You can only enable BGP signaling.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The following sections describe how to configure VPLS multihoming. Some information is also provided on single-homed site configuration versus multihomed site configuration.

VPLS Multihomed Site Configuration

The following describes the requirements for a VPLS multihomed site configuration:

- Assign the same site ID on all PE routers connected to the same CE devices.

  When two PE routers use the same site ID, VPLS assumes multihoming behavior by default. The site preference value is used to signal the primary and backup PE router. In such cases, when multihoming is explicitly configured using the `multi-homing` statement , it is only used for tracking the BGP peer status, such as to prevent isolation of the PE router from the core or split brain. There are scenarios, however that require preventing of BGP peer tracking, such as in a two-PE-router topology. In such cases, multihoming should not be explicitly configured as it can break node redundancy.

  When identical site IDs are used without configuring multihoming, a collision log message is generated at each signaling: `RPD_L2VPN_SITE_COLLISION: Same site ID 2 configured on remote PE (RD 8.8.8.1:1013:) and local PE in VPN 1013 (non-multihomed site 2)`. This is expected behavior.

- Assign unique route distinguishers for each multihomed PE router.

- Reference all interfaces assigned to the multihomed VPLS site on each PE router. Only one of these interfaces is used to send and receive traffic for this site at a time.

- Either designate a primary interface or allow the router to select the interface to be used as the primary interface.

  If the router selects the interface, the interface used to connect the PE router to the site depends on the order in which interfaces are listed in the PE router’s configuration. The first operational interface in the set of configured interfaces is chosen to be the designated interface. If this interface fails, the next interface in the list is selected to send and receive traffic for the site.

The following configuration shows the statements you need to configure to enable VPLS multihoming:

```
[edit routing-instances routing-instance-name]
instance-type vpls;
interface interface-name;
interface interface-name;
protocols vpls {
    site site-name {
        active-interface {
            any;
            primary interface-name;
        }
        interface interface-name;
        interface interface-name;
        multi-homing;
        site-identifier number;
    }
}
route-distinguisher (as-number:id | ip-address:id);
```

NOTE: If you add a direct connection between CE devices that are multihomed to the same VPLS site on different PE routers, the traffic can loop and a loss of connectivity might occur. We do not recommend this topology.

Most of these statements are explained in more detail in the rest of this chapter. The following sections explain how to configure the statements that are specific to VPLS multihoming:

Specifying an Interface as the Active Interface

You need to specify one of the interfaces for the multihomed site as the primary interface. If there are multiple interfaces, the remaining interfaces are activated only when the primary interface goes down. If no active interfaces are configured at the site level, all traffic for a VPLS site travels through a single, non-multihomed PE router.

You must configure one of the following options for the `active-interface` statement:

- `any`—One configured interface is randomly designated as the active interface for the VPLS site.
- `primary`—Specify the name of the multihomed interface to be used as the primary interface by the VPLS site.

To specify a multihomed interface as the primary interface for the VPLS site, include the `active-interface` statement:

```
active-interface {
    any;
    primary interface-name;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls site` *`site-name`*`]`

Configuring Multihoming on the PE Router

When a CE device is connected to the same VPLS site on more than one PE router, including the `multi-homing` statement on all associated PE routers results in tracking of BGP peers. If no BGP peer is available, VPLS deactivates all active interfaces for a site. When two PE routers use the same site ID, VPLS assumes multihoming behavior by default. The site preference value is used to signal the primary and backup PE router. In such cases, when multihoming is explicitly configured using the `multi-homing` statement , it is only used for tracking the BGP peer status, such as to prevent isolation of the PE router from the core or split brain.

NOTE: When identical site IDs are used without configuring multihoming, a collision log message is generated at each signaling: `RPD_L2VPN_SITE_COLLISION: Same site ID 2 configured on remote PE (RD 8.8.8.1:1013:) and local PE in VPN 1013 (non-multihomed site 2)`. This is expected behavior.

VPLS Single-Homed Site Configuration

All VPLS single-homed sites are connected to the same default VE device. All interfaces in a VPLS routing instance that are not configured as part of a multihomed site are assumed to be single-homed to the default VE device.

RELATED DOCUMENTATION

VPLS Multihoming Overview | 885

Example: VPLS Multihoming, Improved Convergence Time | 957

active-interface

multi-homing

Example: VPLS Multihoming, Improved Convergence Time

IN THIS SECTION

- Requirements | 958
- Overview | 958
- Configuration | 960

This example shows how to configure a virtual private LAN service (VPLS) employing multihoming to a customer site. This particular VPLS multihoming example shows how to configure a feature that improves the network convergence time in the event a multihomed site needs to switch traffic to its alternate PE router.

Requirements

This example uses the following hardware and software components:

- Three M Series, MX Series, or T Series routers
- Junos OS Release 12.2 or later

If you are using M Series or T Series routers, the PE routers must have either virtual loopback tunnel (`vt`) interfaces or label-switched interfaces (LSIs). On M Series and T Series routers, VPLS uses tunnel-based PICs to create virtual ports on vt interfaces. If you do not have a tunnel-based PIC installed on your M Series or T Series router, you can still configure VPLS by using LSIs to support the virtual ports. Use of LSIs requires Ethernet-based PICs installed in an Enhanced Flexible PIC Concentrator (FPC).

You do not need to use routers for the CE devices. For example, the CE devices can be EX Series Ethernet Switches.

Overview

IN THIS SECTION

● Topology | 959

All PE routers in a VPLS network operate like a large, distributed Ethernet switch to provide Layer 2 services to attached devices. This example illustrates a network of PE routers and CE devices configured to use VPLS multihoming.. The topology consists of six routers: four PE routers and two CE devices. Device CE1 is multihomed to Routers PE1 and PE2. The PE routers are configured with the `best-site` and `mac-flush` statements to improve the convergence time in the event the connection between Device CE1 and one of its multihomed PE routers fails.

This example includes the following settings:

- `best-site`—Uses the B-bit of the control flags bit vector (the third bit counting from the most significant bit) within the Layer 2 information extended community to indicate that the site is preferred. Each VPLS site configured with the `best-site` statement signals to the other PE routers that it is the preferred site. The Layer 2 information extended community includes the following information:

- Extended community type (2 octets)
- Encapsulation type (1 octet)
- Control flags (1 octet)
- Layer 2 MTU (2 octets)
- Reserved (2 octets)

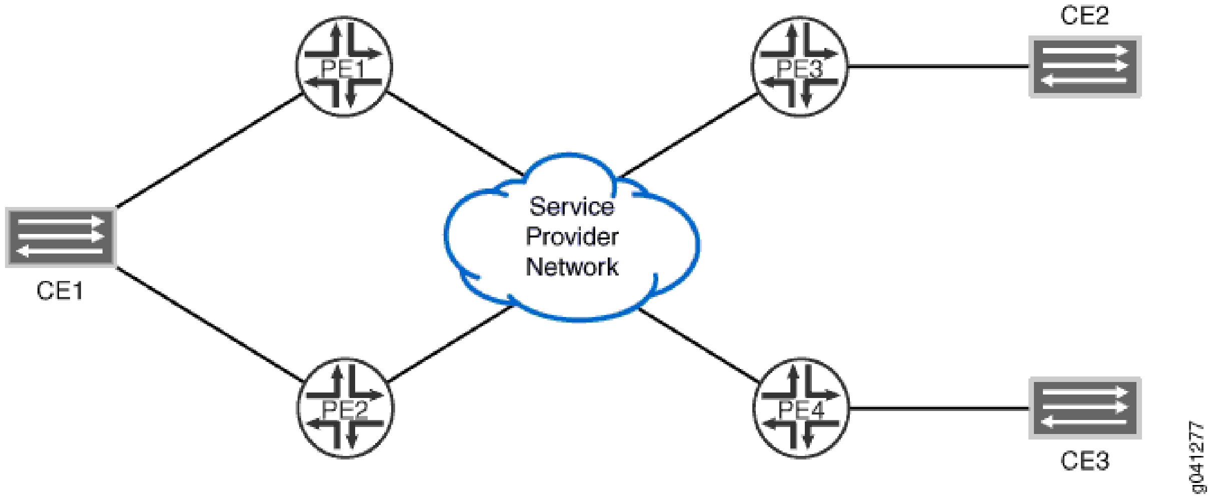
  When a neighboring PE router within the VPLS routing instance receives the label block advertisement, it knows that the corresponding PE router is the most preferable router of those remote PE routers multihomed to that site. If a neighboring PE router does not support the best site feature, the standard local site selection process is used. For example, if Router PE1 does not receive a B-bit from any of the label blocks advertisements received from Router PE3, Router PE1 proceeds to assume that Router PE3 does not support the best site feature. It creates a virtual circuit based on its minimum-designated site. For the other PE routers that do support the best site feature, Router PE1 builds virtual circuits using the locally tagged best site.

- `mac-flush`—Enables media access control (MAC) flush processing for the VPLS routing instance or for the mesh group under a VPLS routing instance. MAC flush processing removes MAC addresses from the MAC address database that have been learned dynamically. With the dynamically learned MAC addresses removed, MAC address convergence requires less time to complete.

Topology

Figure 69 on page 960 shows the topology used in this example. Router PE2 is configured with the `best-site` statement and acts as the preferred gateway for traffic from Device CE1.

Figure 69: VPLS Multihoming Topology with Router PE2 Configured as the Best Site



Configuration

IN THIS SECTION

- CLI Quick Configuration | 960
- Router PE1 | 963
- Router PE2 | 967
- Router PE3 | 971

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 PE1:

```
set interfaces fe-0/1/0 encapsulation ethernet-vpls
set interfaces fe-0/1/0 unit 0 family vpls
set interfaces fe-0/1/2 unit 0 family inet address 10.0.59.14/32
set interfaces fe-0/1/2 unit 0 family iso
set interfaces fe-0/1/2 unit 0 family mpls
set interfaces fe-0/1/3 unit 0 family inet address 10.0.89.14/30
set interfaces fe-0/1/3 unit 0 family iso
set interfaces fe-0/1/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.9.1/32
set interfaces lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5003.00
set routing-options router-id 192.168.9.1
set protocols mpls interface all
set protocols bgp group int type internal
set protocols bgp group int local-address 192.0.2.1
set protocols bgp group int family l2vpn signaling
set protocols isis level 1 disable
set protocols isis interface fe-0/1/2.0
set protocols isis interface fe-0/1/3.0
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface fe-0/1/0.0
set routing-instances vpls_1 route-distinguisher 10.255.107.74:1
set routing-instances vpls_1 vrf-target target:65056:1
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls site site_3 site-identifier 3
set routing-instances vpls_1 protocols vpls site site_3 multi-homing
set routing-instances vpls_1 protocols vpls site site_3 site-preference primary
set routing-instances vpls_1 protocols vpls site site_3 interface fe-0/1/0.0
set routing-instances vpls_1 protocols vpls site site_994 site-identifier 994
set routing-instances vpls_1 protocols vpls mac-flush
```

Router PE2:

```
set interfaces fe-0/1/1 encapsulation ethernet-vpls
set interfaces fe-0/1/1 unit 0 family vpls
set interfaces fe-0/1/2 unit 0 family inet address 10.0.59.13/32
set interfaces fe-0/1/2 unit 0 family iso
set interfaces fe-0/1/2 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.168.5.1/32
set interfaces lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5005.00
set routing-options router-id 192.168.5.1
set protocols mpls interface all
set protocols isis level 1 disable
set protocols isis interface fe-0/1/2.0
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface fe-0/1/1.0
set routing-instances vpls_1 route-distinguisher 10.255.107.76:1
set routing-instances vpls_1 vrf-target target:65056:1
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls site site_3 site-identifier 3
set routing-instances vpls_1 protocols vpls site site_3 multi-homing
set routing-instances vpls_1 protocols vpls site site_3 site-preference backup
set routing-instances vpls_1 protocols vpls site site_3 interface fe-0/1/1.0
set routing-instances vpls_1 protocols vpls site site_995 site-identifier 995
set routing-instances vpls_1 protocols vpls site site_995 best-site
set routing-instances vpls_1 protocols vpls mac-flush
```

Router PE3:

```
set interfaces fe-1/3/0 unit 0 description "PE3 to PE1"
set interfaces fe-1/3/0 unit 0 family inet address 10.0.89.13/30
set interfaces fe-1/3/0 unit 0 family iso
set interfaces fe-1/3/0 unit 0 family mpls
set interfaces fe-1/3/1 encapsulation ethernet-vpls
set interfaces fe-1/3/1 unit 0 family vpls
set interfaces lo0 unit 0 family inet address 192.168.8.1/32
set interfaces lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5002.00
set routing-options router-id 192.168.8.1
set protocols isis level 1 disable
set protocols isis interface fe-1/3/0.0
set protocols isis interface lo0.0
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set protocols mpls interface all
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface fe-1/3/1.0
set routing-instances vpls_1 route-distinguisher 10.255.107.72:1
set routing-instances vpls_1 vrf-target target:65056:1
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls site site_2 site-identifier 2
set routing-instances vpls_1 protocols vpls site site_2 interface fe-0/1/0.100
set routing-instances vpls_1 protocols vpls site site_993 site-identifier 993
set routing-instances vpls_1 protocols vpls mac-flush
```

Router PE1

Step-by-Step Procedure

To configure Router PE1:

1. Configure the interfaces, interface encapsulation, and the protocol families.

   ```
   [edit interfaces]
   user@PE1# set fe-0/1/0 encapsulation ethernet-vpls
   user@PE1# set fe-0/1/0 unit 0 family vpls
   user@PE1# set fe-0/1/2 unit 0 family inet address 10.0.59.14/32
   user@PE1# set fe-0/1/2 unit 0 family iso
   user@PE1# set fe-0/1/2 unit 0 family mpls
   user@PE1# set fe-0/1/3 unit 0 family inet address 10.0.89.14/30
   user@PE1# set fe-0/1/3 unit 0 family iso
   user@PE1# set interfaces fe-0/1/3 unit 0 family mpls
   user@PE1# set lo0 unit 0 family inet address 192.168.9.1/32
   user@PE1# set lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5003.00
   ```

2. Configure the protocol-independent properties.

   ```
   [edit routing-options]
   user@PE1# set router-id 192.168.9.1
   ```

3. Configure MPLS on the router’s interfaces.

   ```
   [edit protocols mpls]
   user@PE1# set interface all
   ```

4. Configure BGP.

   ```
   [edit protocols bgp]
   user@PE1# set group int type internal
   ```

```
user@PE1# set group int local-address 192.0.2.1
user@PE1# set group int family l2vpn signaling
```

5. Configure IS-IS as the IGP between the PE routers.

```
[edit protocols isis]
user@PE1# set level 1 disable
user@PE1# set interface fe-0/1/3.0
user@PE1# set interface lo0.0
```

6. Configure LDP as the signaling protocol for MPLS.

```
[edit protocols ldp]
user@PE1# set interface all
user@PE1# set interface fxp0.0 disable
user@PE1# set interface lo0.0
```

7. Configure the VPLS routing instance.

   Include the `mac-flush` statement to ensure that stale routes are removed from Router PE1 promptly.

```
[edit routing-instances vpls_1]
user@PE1# set instance-type vpls
user@PE1# set interface fe-0/1/0.0
user@PE1# set route-distinguisher 10.255.107.74:1
user@PE1# set vrf-target target:65056:1
user@PE1# set protocols vpls no-tunnel-services
user@PE1# set protocols vpls site site_3 site-identifier 3
user@PE1# set protocols vpls site site_3 multi-homing
user@PE1# set protocols vpls site site_3 site-preference primary
user@PE1# set protocols vpls site site_3 interface fe-0/1/0.0
user@PE1# set protocols vpls site site_994 site-identifier 994
user@PE1# set protocols vpls mac-flush
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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@PE1# show interfaces
fe-0/1/0 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
    fe-0/1/2 {
        unit 0 {
            family inet {
                address 10.0.59.14/32;
            }
            family iso;
            family mpls;
        }
    }
fe-0/1/3 {
    unit 0 {
        family inet {
            address 10.0.89.14/30;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.9.1/32;
        }
        family iso {
            address 47.0005.8083.0000.1921.6800.5003.00;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface all;
}
bgp {
    group int {
        type internal;
        local-address 192.0.2.1;
        family l2vpn {
            signaling;
        }
    }
}
isis {
    level 1 disable;
    interface fe-0/1/2.0;
    interface fe-0/1/3.0;
    interface lo0.0;
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@PE1# show routing-instances
vpls_1 {
    instance-type vpls;
    interface fe-0/1/0.0;
    route-distinguisher 10.255.107.74:1;
    vrf-target target:65056:1;
    protocols {
        vpls {
            no-tunnel-services;
            site site_3 {
                site-identifier 3;
                multi-homing;
                site-preference primary;
                interface fe-0/1/0.0;
            }
            site site_994 {
                site-identifier 994;
            }
            mac-flush;
        }
    }
}
```

```
user@PE1# show routing-options
router-id 192.168.9.1;
```

Router PE2

Step-by-Step Procedure

To configure Router PE2:

1. Configure the interfaces, interface encapsulation, and the protocol families.

```
[edit interfaces]
user@PE2# set fe-0/1/1 encapsulation ethernet-vpls
user@PE2# set fe-0/1/1 unit 0 family vpls
user@PE2# set fe-0/1/2 unit 0 family inet address 10.0.59.13/32
user@PE2# set fe-0/1/2 unit 0 family iso
user@PE2# set fe-0/1/2 unit 0 family mpls
user@PE2# set lo0 unit 0 family inet address 192.168.5.1/32
user@PE2# set lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5005.00
```

2. Configure the protocol-independent properties.

```
[edit routing-options]
user@PE2# set router-id 192.168.5.1
```

3. Configure MPLS on the Router PE2 interfaces.

```
[edit protocols]
user@PE2# set mpls interface all
```

4. Configure the LDP as the signaling protocol for MPLS on the PE router facing interface.

```
[edit protocols ldp]
user@PE2# set interface all
user@PE2# set interface fxp0.0 disable
user@PE2# set interface lo0.0
```

5. Configure IS-IS as the IGP between the PE routers.

```
[edit protocols isis]
user@PE2# set level 1 disable
user@PE2# set interface fe-0/1/2.0
user@PE2# set interface lo0.0
```

6. Configure the VPLS routing instance vpls_1.

   Include the `best-site` statement to ensure that Router PE2 acts as the preferred path for the CE router. Include the `mac-flush` statement to ensure that stale routes are removed from Router PE2 promptly.

```
[edit routing-instances vpls_1]
user@PE2# set instance-type vpls
user@PE2# set interface fe-0/1/1.0
user@PE2# set route-distinguisher 10.255.107.76:1
user@PE2# set vrf-target target:65056:1
user@PE2# set protocols vpls no-tunnel-services
user@PE2# set protocols vpls site site_3 site-identifier 3
user@PE2# set protocols vpls site site_3 multi-homing
user@PE2# set protocols vpls site site_3 site-preference backup
user@PE2# set protocols vpls site site_3 interface fe-0/1/1.0
user@PE2# set protocols vpls site site_995 site-identifier 995
user@PE2# set protocols vpls site site_995 best-site
user@PE2# set protocols vpls mac-flush
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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@PE2# show interfaces
fe-0/1/1 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
fe-0/1/2 {
    unit 0 {
        family inet {
            address 10.0.59.13/32;
        }
        family iso;
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.5.1/32;
        }
        family iso {
            address 47.0005.8083.0000.1921.6800.5005.00;
        }
    }
}
```

```
user@PE2# show protocols
mpls {
    interface all;
}
isis {
    level 1 disable;
    interface fe-0/1/2.0;
    interface lo0.0;
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@PE2# show routing-instances
vpls_1 {
    instance-type vpls;
    interface fe-0/1/1.0;
    route-distinguisher 10.255.107.76:1;
    vrf-target target:65056:1;
    protocols {
        vpls {
            no-tunnel-services;
            site site_3 {
                site-identifier 3;
                multi-homing;
                site-preference backup;
                interface fe-0/1/1.0;
            }
            site site_995 {
                site-identifier 995;
                best-site;
            }
            mac-flush;
        }
    }
}
```

```
user@pe2# show routing-options
router-id 192.168.5.1;
```

Router PE3

Step-by-Step Procedure

To configure Router PE3:

1. Configure the interfaces, interface encapsulation, and the protocol families.

```
[edit interfaces]
user@PE3# set fe-1/3/0 unit 0 description "PE3 to PE1"
user@PE3# set fe-1/3/0 unit 0 family inet address 10.0.89.13/30
user@PE3# set fe-1/3/0 unit 0 family iso
user@PE3# set fe-1/3/0 unit 0 family mpls
user@PE3# set fe-1/3/1 encapsulation ethernet-vpls
user@PE3# set fe-1/3/1 unit 0 family vpls
user@PE3# set lo0 unit 0 family inet address 192.168.8.1/32
user@PE3# set lo0 unit 0 family iso address 47.0005.8083.0000.1921.6800.5002.00
```

2. Configure the protocol-independent properties.

```
[edit routing-options]
user@PE3# set router-id 192.168.8.1
```

3. Configure IS-IS as the IGP between the PE routers.

```
[edit protocols isis]
user@PE3# set level 1 disable
user@PE3# set interface fe-0/1/3.0
user@PE3# set interface lo0.0
```

4. Configure LDP as the signaling protocol for MPLS.

```
[edit protocols ldp]
user@PE3# set interface all
user@PE3# set interface fxp0.0 disable
user@PE3# set interface lo0.0
```

5. Configure the VPLS routing instance.

Include the `mac-flush` statement here to ensure that stale routes are removed from Router PE1 promptly.

```
[edit routing-instances vpls_1]
user@PE3# set instance-type vpls
user@PE3# set interface fe-1/3/1.0
user@PE3# set route-distinguisher 10.255.107.72:1
user@PE3# set vrf-target target:65056:1
user@PE3# set protocols vpls no-tunnel-services
user@PE3# set protocols vpls site site_2 site-identifier 2
user@PE3# set protocols vpls site site_2 interface fe-0/1/0.100
user@PE3# set protocols vpls site site_993 site-identifier 993
user@PE3# set protocols vpls mac-flush
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols` `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@PE3# show interfaces
fe-1/3/0 {
    unit 0 {
        description "PE3 to PE1";
        family inet {
            address 10.0.89.13/30;
        }
        family iso;
        family mpls;
    }
}
fe-1/3/1 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.8.1/32;
        }
        family iso {
            address 47.0005.8083.0000.1921.6800.5002.00;
        }
    }
}
```

```
user@PE3# show protocols
mpls {
    interface all;
}
bgp {
    group int {
        type internal;
        local-address 192.0.2.2;
        family l2vpn {
            signaling;
        }
    }
}
isis {
    level 1 disable;
    interface fe-1/3/0.0;
    interface lo0.0;
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
```

```
user@PE3# show routing-instances
vpls_1 {
    instance-type vpls;
    interface fe-0/1/0.100; ## 'fe-0/1/0.100' is not defined
    route-distinguisher 10.255.107.72:1;
    vrf-target target:65056:1;
    protocols {
        vpls {
            no-tunnel-services;
            site site_2 {
                site-identifier 2;
                interface fe-1/3/1.0;
            }
            site site_993 {
                site-identifier 993;
            }
            mac-flush;
        }
    }
}
```

```
user@pe3# show routing-options
router-id 192.168.8.1;
```

RELATED DOCUMENTATION

Configuring VPLS Multihoming | **954**

Example: Configuring VPLS Multihoming (FEC 129) | **974**

best-site

multi-homing

Example: Configuring VPLS Multihoming (FEC 129)

IN THIS SECTION

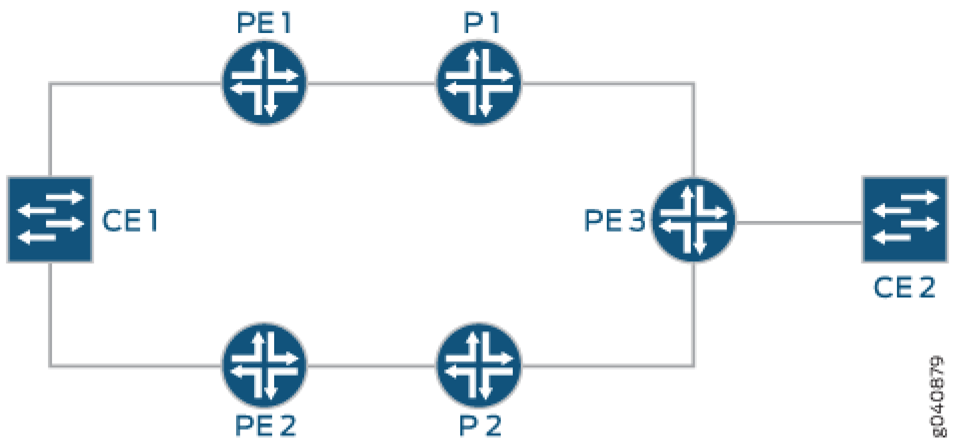
- VPLS Multihoming Overview | **975**
- Example: Configuring VPLS Multihoming (FEC 129) | **977**

VPLS Multihoming Overview

Virtual private LAN service (VPLS) multihoming enables you to connect a customer site to two or more PE routers to provide redundant connectivity. A redundant PE router can provide network service to the customer site as soon as a failure is detected. VPLS multihoming helps to maintain VPLS service and traffic forwarding to and from the multihomed site in the event of the following types of network failures:

- PE router to CE device link failure
- PE router failure
- MPLS-reachability failure between the local PE router and a remote PE router

Figure 70: CE Device Multihomed to Two PE Routers



NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Figure 70 on page 975 illustrates how a CE device could be multihomed to two PE routers. Device CE1 is multihomed to Routers PE1 and PE2. Device CE2 has two potential paths to reach Device CE1, but only one path is active at any one time. If Router PE1 were the designated VPLS edge (VE) device (also called a designated forwarder), BGP would signal a pseudowire from Router PE3 to Router PE1. If a failure occurred over this path, Router PE2 would be made the designated VE device, and BGP would re-signal the pseudowire from Router PE3 to Router PE2.

Multihomed PE routers advertise network layer reachability information (NLRI) for the multihomed site to the other PE routers in the VPLS network. The NLRI includes the site ID for the multihomed PE routers. For all of the PE routers multihomed to the same CE device, you need to configure the same site ID. The remote VPLS PE routers use the site ID to determine where to forward traffic addressed to the customer site. To avoid route collisions, the site ID shared by the multihomed PE routers must be different than the site IDs configured on the remote PE routers in the VPLS network.

Although you configure the same site ID for each of the PE routers multihomed to the same CE device, you can configure unique values for other parameters, such as the route distinguisher. These values help to determine which multihomed PE router is selected as the designated VE device to be used to reach the customer site.

BEST PRACTICE: We recommend that you configure unique route distinguishers for each multihomed PE router. Configuring unique route distinguishers helps with faster convergence when the connection to a primary multihomed PE router goes down. If you configure unique route distinguishers, the other PE routers in the VPLS network must maintain additional state for the multihomed PE routers.

Remote PE routers in the VPLS network need to determine which of the multihomed PE routers should forward traffic to reach the CE device. To make this determination, remote PE routers use the VPLS path-selection process to select one of the multihomed PE routers based on its NLRI advertisement. Because remote PE routers pick only one of the NLRI advertisements, it establishes a pseudowire to only one of the multihomed PE routers, the PE router that originated the winning advertisement. This prevents multiple paths from being created between sites in the network, preventing the formation of Layer 2 loops. If the selected PE router fails, all PE routers in the network automatically switch to the backup PE router and establish new pseudowires to it.

BEST PRACTICE: To prevent the formation of Layer 2 loops between the CE devices and the multihomed PE routers, we recommend that you employ the Spanning Tree Protocol (STP) on your CE devices. Layer 2 loops can form due to incorrect configuration. Temporary Layer 2 loops can also form during convergence after a change in the network topology.

The PE routers run the BGP path selection procedure on locally originated and received Layer 2 route advertisements to establish that the routes are suitable for advertisement to other peers, such as BGP route reflectors. If a PE router in a VPLS network is also a route reflector, the path selection process for the multihomed site has no effect on the path selection process performed by this PE router for the purpose of reflecting Layer 2 routes. Layer 2 prefixes that have different route distinguishers are considered to have different NLRIs for route reflection. The VPLS path selection process enables the route reflector to reflect all routes that have different route distinguishers to the route reflector clients, even though only one of these routes is used to create the VPLS pseudowire to the multihomed site.

Junos OS supports VPLS multihoming for both BGP VPLS and FEC129 VPLS. See Feature Explorer for information on platform and release support for any VPLS-related feature.

SEE ALSO

Configuring VPLS Multihoming | 954
Advantages of Using Autodiscovery for VPLS Multihoming | 888
Example: Configuring VPLS Multihoming (FEC 129) | 977
Example: VPLS Multihoming, Improved Convergence Time | 957

Example: Configuring VPLS Multihoming (FEC 129)

IN THIS SECTION

- Requirements | 977
- Overview | 977
- Configuration | 979
- Verification | 989

This example shows how to configure virtual private LAN service (VPLS) multihoming. Multihoming allows a customer site to connect to multiple provider edge (PE) routers. A VPLS site multihomed to two or more PE routers provides redundant connectivity in the event of a PE router-to-CE device link failure or the failure of a PE router. The example demonstrates BGP-based multihoming support for FEC 129 VPLS (also known as LDP VPLS with BGP-based autodiscovery).

Requirements

This example has the following hardware and software requirements:

- One or more CE devices to represent a VPLS site.
- Two or more PE devices.
- Junos OS Release 12.3 or later running on the PE devices that are connected to the multihomed VPLS site.

Overview

BGP-based VPLS autodiscovery (FEC 129) enables each VPLS PE router to discover the other PE routers that are in the same VPLS domain. VPLS autodiscovery also automatically detects when PE routers are added or removed from the VPLS domain. You do not need to manually configure the VPLS and maintain the configuration when a PE router is added or deleted. VPLS autodiscovery uses BGP to discover the VPLS members and to set up and tear down pseudowires in the VPLS.

BGP multihoming enables you to connect a customer site to two or more PE routers to provide redundant connectivity while preventing the formation of Layer 2 loops in the service provider's network. The redundant connectivity maintains the VPLS service and traffic forwarding to and from the multihomed site in the event of a PE router-to-CE device link failure, the failure of a PE router, or an MPLS reachability failure between the local PE router and a remote PE router. A redundant PE router can begin providing service to the customer site as soon as the failure is detected.

When a CE device connects to multiple PE routers, each of these routers advertises reachability for the multihomed site—routes that have the same site ID in the Layer 2 network layer reachability information (NLRI). The other PE routers in the network use a BGP path selection process to select only one of the advertising routers to which they send traffic destined for the CE device. This path selection process eliminates Layer 2 loops in the VPLS network.

Autodiscovery is not specifically related to multihoming. Autodiscovery is not required for multihoming to work. They are two separate features. That said, the meaning of FEC 129 is that VPLS does autodiscovery. So when you configure multihoming for FEC 129, you must also, by definition, configure autodiscovery (with the `auto-discovery-only` statement).

There are two places in the configuration where you can configure VPLS multihoming. One is for FEC 128, and the other is for FEC 129:

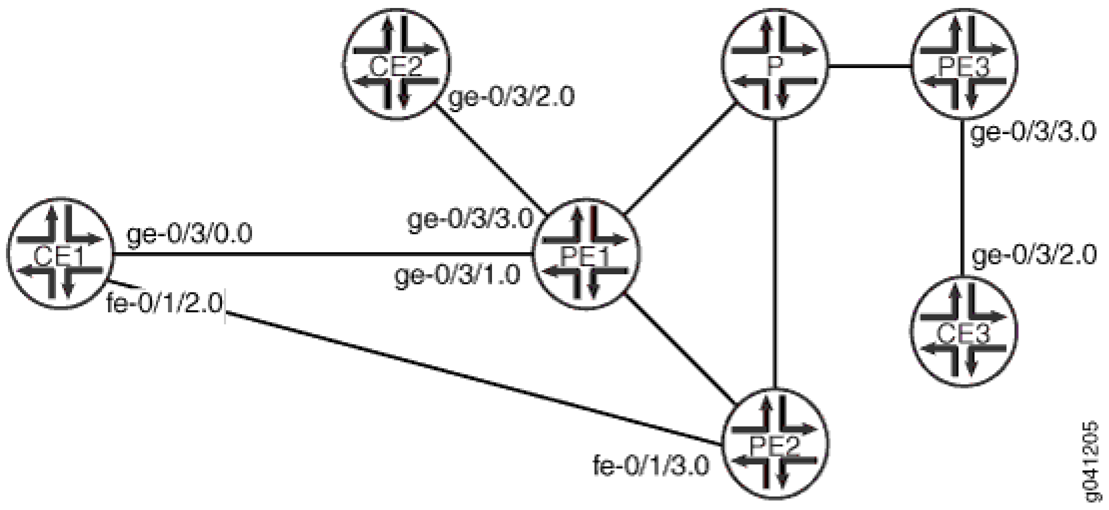
- For FEC 128—`routing-instances` *`instance-name`* `protocols vpls site` *`site-name`* `multi-homing`
- For FEC 129—`routing-instances` *`instance-name`* `protocols vpls multi-homing`

The following statements are used for configuring multihoming for FEC 129:

```
[edit routing-instances instance-name protocols vpls]
multi-homing {
    peer-active;
    site site-name {
        active-interface interface-name {
            any;
            primary interface-name;
        }
        identifier identifier;
        interface interface-name {
            preference preference-value;
        }
        peer-active;
        preference (preference-value | backup | primary);
    }
}
```

This example shows Device CE1 multihomed to Router PE1 and Router PE2. In addition, Device CE2 is single-homed to Router PE1. Device PE3 is the remote PE router, connected to Device CE3. Multihoming is not enabled on Device PE3. "CLI Quick Configuration" on page 979 shows the configuration for all of the devices in Figure 71 on page 979. The section "Configuring Device PE1" on page 984 has step-by-step instructions for configuring Device PE1.

Figure 71: Topology for FEC 129 Multihoming



Configuration

IN THIS SECTION

- CLI Quick Configuration | 979
- Configuring Device PE1 | 984

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 PE1

```
set interfaces ge-0/3/3 encapsulation ethernet-vpls
set interfaces ge-0/3/3 unit 0 description PE1-to-CE2
set interfaces ge-0/3/3 unit 0 family vpls
set interfaces ge-0/3/1 encapsulation ethernet-vpls
set interfaces ge-0/3/1 unit 0 description PE1-to-CE1
set interfaces ge-0/3/1 unit 0 family vpls
set interfaces ge-1/2/0 unit 0 description PE1-to-P
set interfaces ge-1/2/0 unit 0 family inet address 10.1.1.1/30
set interfaces ge-1/2/0 unit 0 family mpls
set interfaces ge-1/2/1 unit 0 description PE1-to-PE2
set interfaces ge-1/2/1 unit 0 family inet address 10.1.1.5/30
set interfaces ge-1/2/1 unit 0 family mpls
set interfaces lo0 unit 2 family inet address 192.0.2.2/24
set protocols mpls interface ge-1/2/0.0
set protocols mpls interface ge-1/2/1.0
set protocols bgp local-address 192.0.2.2
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.3
set protocols bgp group pe-pe neighbor 192.0.2.4
set protocols bgp group pe-pe neighbor 192.0.2.5
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-1/2/0.0
set protocols ospf area 0.0.0.0 interface ge-1/2/1.0
set protocols ospf area 0.0.0.0 interface lo0.2 passive
set protocols ldp interface ge-1/2/0.0
set protocols ldp interface ge-1/2/1.0
set protocols ldp interface lo0.2
set routing-instances green instance-type vpls
set routing-instances green interface ge-0/3/1.0
set routing-instances green interface ge-0/3/3.0
set routing-instances green route-distinguisher 192.0.2.2:1
set routing-instances green l2vpn-id l2vpn-id:100:100
set routing-instances green vrf-target target:100:100
set routing-instances green protocols vpls no-tunnel-services
set routing-instances green protocols vpls oam ping-interval 600
set routing-instances green protocols vpls oam bfd-liveness-detection minimum-interval 200
set routing-instances green protocols vpls multi-homing site test identifier 1
set routing-instances green protocols vpls multi-homing site test interface ge-0/3/1.0
set routing-options router-id 192.0.2.2
set routing-options autonomous-system 100
```

Device PE2

```
set interfaces fe-0/1/3 encapsulation ethernet-vpls
set interfaces fe-0/1/3 unit 0 description PE2-to-CE1
set interfaces fe-0/1/3 unit 0 family vpls
set interfaces ge-1/2/0 unit 0 description PE2-to-PE1
set interfaces ge-1/2/0 unit 0 family inet address 10.1.1.6/30
set interfaces ge-1/2/0 unit 0 family mpls
set interfaces ge-1/2/2 unit 0 description PE2-to-P
set interfaces ge-1/2/2 unit 0 family inet address 10.1.1.10/30
set interfaces ge-1/2/2 unit 0 family mpls
set interfaces lo0 unit 4 family inet address 192.0.2.4/32
set protocols mpls interface ge-1/2/0.0
set protocols mpls interface ge-1/2/2.0
set protocols bgp local-address 192.0.2.4
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols bgp group pe-pe neighbor 192.0.2.3
set protocols bgp group pe-pe neighbor 192.0.2.5
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-1/2/0.0
set protocols ospf area 0.0.0.0 interface ge-1/2/2.0
set protocols ospf area 0.0.0.0 interface lo0.4 passive
set protocols ldp interface ge-1/2/0.0
set protocols ldp interface ge-1/2/2.0
set protocols ldp interface lo0.4
set routing-instances green instance-type vpls
set routing-instances green interface fe-0/1/3.0
set routing-instances green route-distinguisher 192.0.2.4:1
set routing-instances green l2vpn-id l2vpn-id:100:100
set routing-instances green vrf-target target:100:100
set routing-instances green protocols vpls no-tunnel-services
set routing-instances green protocols vpls oam ping-interval 600
set routing-instances green protocols vpls oam bfd-liveness-detection minimum-interval 200
set routing-instances green protocols vpls multi-homing site test identifier 1
set routing-instances green protocols vpls multi-homing site test interface fe-0/1/3.0
set routing-options router-id 192.0.2.4
set routing-options autonomous-system 100
```

Device PE3

```
set interfaces ge-0/3/3 unit 0 encapsulation ethernet-vpls
set interfaces ge-0/3/3 unit 0 description PE3-to-CE3
set interfacesge-0/3/3 unit 0 family vpls
set interfaces ge-1/2/0 unit 0 description PE3-to-P
set interfaces ge-1/2/0 unit 0 family inet address 10.1.1.14/30
set interfaces ge-1/2/0 unit 0 family mpls
set interfaces lo0 unit 5 family inet address 192.0.2.5/24
set protocols rsvp interface ge-1/2/0.0
set protocols mpls interface ge-1/2/0.0
set protocols bgp local-address 192.0.2.5
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn auto-discovery-only
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols bgp group pe-pe neighbor 192.0.2.3
set protocols bgp group pe-pe neighbor 192.0.2.4
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.5 passive
set protocols ldp interface ge-1/2/0.0
set protocols ldp interface lo0.5
set routing-instances green instance-type vpls
set routing-instances green interface ge-0/3/3.0
set routing-instances green route-distinguisher 192.0.2.5:100
set routing-instances green l2vpn-id l2vpn-id:100:100
set routing-instances green vrf-target target:100:100
set routing-instances green protocols vpls no-tunnel-services
set routing-instances green protocols vpls oam ping-interval 600
set routing-instances green protocols vpls oam bfd-liveness-detection minimum-interval 200
set routing-instances green protocols vpls oam ping-interval 600
set routing-instances green protocols vpls oam bfd-liveness-detection minimum-interval 200
set routing-options router-id 192.0.2.5
set routing-options autonomous-system 100
```

Device CE1

```
set interfaces ge-0/3/0 unit 0 description CE1-to-PE1
set interfaces ge-0/3/0 unit 0 family inet address 192.0.2.15/24
set interfaces fe-0/1/2 unit 0 description CE1-to-PE2
set interfaces fe-0/1/2 unit 0 family inet address 192.0.2.11/24
```

Device CE2

```
set interfaces ge-0/3/2 unit 0 description CE2-to-PE1
set interfaces ge-0/3/2 unit 0 family inet address 192.0.2.16/24
```

Device CE3

```
set interfaces ge-0/3/2 unit 0 description CE3-to-PE3
set interfaces ge-0/3/2 unit 0 family inet address 192.0.2.17/24
```

Device P

```
set interfaces ge-1/2/0 unit 0 description P-to-PE1
set interfaces ge-1/2/0 unit 0 family inet address 10.1.1.2/30
set interfaces ge-1/2/0 unit 0 family mpls
set interfaces ge-3/2/0 unit 0 description P-to-PE2
set interfaces ge-3/2/0 unit 0 family inet address 10.1.1.9/30
set interfaces ge-3/2/0 unit 0 family mpls
set interfaces ge-4/2/0 unit 0 description P-to-PE3
set interfaces ge-4/2/0 unit 0 encapsulation ethernet
set interfaces ge-4/2/0 unit 0 family inet address 10.1.1.13/30
set interfaces ge-4/2/0 unit 0 family mpls
set interfaces lo0 unit 3 family inet address 192.0.2.3/32
set protocols mpls interface ge-1/2/0.0
set protocols mpls interface ge-3/2/0.0
set protocols mpls interface ge-4/2/0.0
set protocols bgp local-address 192.0.2.3
set protocols bgp group pe-pe type internal
set protocols bgp group pe-pe family l2vpn signaling
set protocols bgp group pe-pe neighbor 192.0.2.2
set protocols bgp group pe-pe neighbor 192.0.2.4
set protocols bgp group pe-pe neighbor 192.0.2.5
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface ge-1/2/0.0
set protocols ospf area 0.0.0.0 interface ge-3/2/0.0
set protocols ospf area 0.0.0.0 interface ge-4/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.3 passive
set protocols ldp interface ge-1/2/0.0
set protocols ldp interface ge-3/2/0.0
set protocols ldp interface ge-4/2/0.0
set protocols ldp interface lo0.3
set routing-options router-id 192.0.2.3
set routing-options autonomous-system 100
```

Configuring Device PE1

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 Device PE1:

1. Configure the interfaces.

   Configure `family mpls` on the provider-facing interfaces. Configure `family vpls` on the customer-facing interfaces.

   ```
   [edit interfaces]
   user@PE1# set ge-0/3/3 encapsulation ethernet-vpls
   user@PE1# set ge-0/3/3 unit 0 description PE1-to-CE2
   user@PE1# set ge-0/3/3 unit 0 family vpls
   user@PE1# set ge-0/3/1 encapsulation ethernet-vpls
   user@PE1# set ge-0/3/1 unit 0 description PE1-to-CE1
   user@PE1# set ge-0/3/1 unit 0 family vpls
   user@PE1# set ge-1/2/0 unit 0 description PE1-to-P
   user@PE1# set ge-1/2/0 unit 0 family inet address 10.1.1.1/30
   user@PE1# set ge-1/2/0 unit 0 family mpls
   user@PE1# set ge-1/2/1 unit 0 description PE1-to-PE2
   user@PE1# set ge-1/2/1 unit 0 family inet address 10.1.1.5/30
   user@PE1# set ge-1/2/1 unit 0 family mpls
   user@PE1# set lo0 unit 2 family inet address 192.0.2.2/24
   ```

2. Configure the interior gateway protocol (IGP) and signaling protocols on the provider-facing interfaces.

   The `traffic-engineering` statement enables OSPF to advertise the label-switched path (LSP) metric in summary link-state advertisements (LSAs).

   ```
   [edit protocols]
   user@PE1# set ldp interface ge-1/2/0.0
   user@PE1# set ldp interface ge-1/2/1.0
   user@PE1# set ldp interface lo0.2
   user@PE1# set mpls interface ge-1/2/0.0
   user@PE1# set mpls interface ge-1/2/1.0
   user@PE1# set ospf traffic-engineering
   user@PE1# set ospf area 0.0.0.0 interface ge-1/2/0.0
   user@PE1# set ospf area 0.0.0.0 interface ge-1/2/1.0
   user@PE1# set ospf area 0.0.0.0 interface lo0.2 passive
   ```

3. Configure BGP.

   The `auto-discovery-only` statement notifies the routing process (rpd) to expect autodiscovery-related NLRI messages so that information can be deciphered and used by LDP and VPLS. The `auto-discovery-only` statement must be configured on all PE routers in a VPLS. If you configure route reflection, the `auto-discovery-only` statement is also required on provider (P) routers that act as the route reflector in supporting FEC 129-related updates.

   For interoperation scenarios in which a PE router must support both types of NLRI (FEC 128 and FEC 129), this example also includes the `signaling` statement.

   ```
   [edit protocols bgp]
   user@PE1# set local-address 192.0.2.2
   user@PE1# set group pe-pe type internal
   user@PE1# set group pe-pe family l2vpn auto-discovery-only
   user@PE1# set group pe-pe family l2vpn signaling
   user@PE1# set group pe-pe neighbor 192.0.2.3
   user@PE1# set group pe-pe neighbor 192.0.2.4
   user@PE1# set group pe-pe neighbor 192.0.2.5
   ```

4. Configure the routing instance.

   Both CE-facing interfaces are included in the routing instance. Only the multihomed interface is included in the multihoming site.

As a convention, the route distinguisher is composed of Device PE1's loopback interface address and the multihoming site identifier.

```
[edit routing-instances green]
user@PE1# set instance-type vpls
user@PE1# set interface ge-0/3/1.0
user@PE1# set interface ge-0/3/3.0
user@PE1# set route-distinguisher 192.0.2.2:1
user@PE1# set l2vpn-id l2vpn-id:100:100
user@PE1# set vrf-target target:100:100
user@PE1# set protocols vpls no-tunnel-services
user@PE1# set protocols vpls multi-homing site test identifier 1
user@PE1# set protocols vpls multi-homing site test interface ge-0/3/1.0
```

5. (Optional) Configure bidirectional forwarding detection (BFD) for FEC 129 VPLS.

```
[edit routing-instances green]
user@PE1# set protocols vpls oam ping-interval 600
user@PE1# set protocols vpls oam bfd-liveness-detection minimum-interval 200
```

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

```
[edit routing-options]
user@PE1# set router-id 192.0.2.2
user@PE1# set autonomous-system 100
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `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@PE1# show interfaces
ge-0/3/1 {
    encapsulation ethernet-vpls;
    unit 0 {
        description PE1-to-CE1;
        family vpls;
    }
}
ge-0/3/3 {
    encapsulation ethernet-vpls;
    unit 0 {
        description PE1-to-CE2;
        family vpls;
    }
}
ge-1/2/0 {
    unit 0 {
        description PE1-to-P;
        family inet {
            address 10.1.1.1/30;
        }
        family mpls;
    }
}
ge-1/2/1 {
    unit 0 {
        description PE1-to-PE2;
        family inet {
            address 10.1.1.5/30;
        }
        family mpls;
    }
}
lo0 {
    unit 2 {
        family inet {
            address 192.0.2.2/24;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface ge-1/2/0.0;
    interface ge-1/2/1.0;
}
bgp {
    local-address 192.0.2.2;
    group pe-pe {
        type internal;
        family l2vpn {
            auto-discovery-only;
            signaling;
        }
        neighbor 192.0.2.3;
        neighbor 192.0.2.4;
        neighbor 192.0.2.5;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-1/2/0.0;
        interface ge-1/2/1.0;
        interface lo0.2 {
            passive;
        }
    }
}
ldp {
    interface ge-1/2/0.0;
    interface ge-1/2/1.0;
    interface lo0.2;
}
```

```
user@PE1# show routing-instances
green {
    instance-type vpls;
    interface ge-0/3/1.0;
    interface ge-0/3/3.0;
    route-distinguisher 192.0.2.2:100;
    l2vpn-id l2vpn-id:100:100;
    vrf-target target:100:100;
    protocols {
        vpls {
            no-tunnel-services;
            oam {
                ping-interval 600;
                bfd-liveness-detection {
                    minimum-interval 200;
                }
            }
            multi-homing {
                site test {
                    identifier 1;
                    interface ge-0/3/1.0;
                }
            }
        }
    }
}
```

```
user@PE1# show routing-options
router-id 192.0.2.2;
autonomous-system 100;
```

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

Verification

IN THIS SECTION

- Verifying That Multihoming Is Operational | 989
- Checking the Multihoming Routes | 991
- Checking the BFD Sessions | 992
- Pinging the Remote PE Router in the VPLS Domain | 993

Confirm that the configuration is working properly.

Verifying That Multihoming Is Operational

Purpose

Verify that multihoming is operational.

Action

From operational mode, enter the `show vpls connections extensive` command.

```
user@PE1> show vpls connections extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Legend for interface status
Up -- operational
Dn -- down

Instance: green
  L2vpn-id: 100:100
  Local-id: 192.0.2.2
    Number of local interfaces: 2
    Number of local interfaces up: 2
    ge-0/3/1.0
    ge-0/3/3.0
    lsi.101711873                Intf - vpls green local-id 192.0.2.2 remote-id 192.0.2.4
neighbor 192.0.2.4
    Remote-id                 Type  St     Time last up          # Up trans
    192.0.2.4                  rmt   Up     Jan 31 13:49:52 2012           1
```

```
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 262146, Outgoing label: 262146
      Local interface: lsi.101711873, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls green local-id 192.0.2.2 remote-id 192.0.2.4 neighbor 192.0.2.4
    Connection History:
        Jan 31 13:49:52 2012  status update timer
        Jan 31 13:49:52 2012  PE route changed
        Jan 31 13:49:52 2012  Out lbl Update                    262146
        Jan 31 13:49:52 2012  In lbl Update                     262146
        Jan 31 13:49:52 2012  loc intf up               lsi.101711873
  Multi-home:
    Local-site                Id     Pref   State
    test                      1      100    Up
      Number of interfaces: 1
      Number of interfaces up: 1
        ge-0/3/1.0
      Received multi-homing advertisements:
        Remote-PE         Pref   flag   Description
        192.0.2.4           100    0x0
```

Meaning

The output shows the status of multihoming for routing instance green.

Checking the Multihoming Routes

Purpose

Verify that the expected routes are identified as multihoming.

Action

From operational mode, enter the `show route table bgp.l2vpn.0` and `show route table green.l2vpn.0` commands.

```
user@PE1> show route table bgp.l2vpn.0
bgp.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.4:100:192.0.2.4/96 AD
                   *[BGP/170] 1d 03:10:45, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
```

```
                    >    via ge-1/2/1.5
192.0.2.4:100:1:0/96 MH
                   *[BGP/170] 1d 03:10:45, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    >    via ge-1/2/1.5
```

```
user@PE1> show route table green.l2vpn.0
green.l2vpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.2:100:192.0.2.2/96 AD
                   *[VPLS/170] 1d 03:11:03, metric2 1
                         Indirect
192.0.2.4:100:192.0.2.4/96 AD
                   *[BGP/170] 1d 03:11:02, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    >    via ge-1/2/1.5
192.0.2.2:100:1:0/96 MH
                   *[VPLS/170] 1d 03:11:03, metric2 1
                         Indirect
192.0.2.4:100:1:0/96 MH
                   *[BGP/170] 1d 03:11:02, localpref 100, from 192.0.2.4
                      AS path: I, validation-state: unverified
                    >    via ge-1/2/1.5
192.0.2.4:NoCtrlWord:5:100:100:192.0.2.2:192.0.2.4/176
                   *[VPLS/7] 1d 03:11:02, metric2 1
                    >    via ge-1/2/1.5
192.0.2.4:NoCtrlWord:5:100:100:192.0.2.4:192.0.2.2/176
                   *[LDP/9] 1d 03:11:02
                         Discard
```

Meaning

`MH` in the output indicates a multihoming route. `AD` indicates autodiscovery.

Checking the BFD Sessions

Purpose

Verify that the BFD session status is operational.

Action

From operational mode, enter the `show bfd session` command.

```
user@PE1> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
198.51.100.1                Up        ge-1/2/1.0     0.600     0.200        3
198.51.100.1                Up        ge-1/2/0.0     0.600     0.200        3

2 sessions, 2 clients
Cumulative transmit rate 10.0 pps, cumulative receive rate 10.0 pps
```

Meaning

`Up` in the `State` field indicates that BFD is working.

Pinging the Remote PE Router in the VPLS Domain

Purpose

Check the operability of the MPLS Layer 2 virtual private network (VPN) connection.

Action

From operational mode, enter the `ping mpls l2vpn` command with the `fec129` option.

```
user@PE1> ping mpls l2vpn fec129 instance green remote-id 192.0.2.5 remote-pe-address 192.0.2.5
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Meaning

The output shows that the ping operation is successful, meaning that the LSP for a FEC 129 Layer 2 VPN connection is reachable.

SEE ALSO

VPLS Multihoming Overview | 885

RELATED DOCUMENTATION

Example: Configuring BGP Autodiscovery for LDP VPLS | 913

Example: Configuring BGP Autodiscovery for LDP VPLS with User-Defined Mesh Groups | 938

Next-Generation VPLS for Multicast with Multihoming Overview

IN THIS SECTION

- Operation of Next-Generation VPLS for Multicast with Multihoming Using BGP | 995
- Implementation of Redundancy Using VPLS Multihomed Links Between PE and CE Devices | 999

VPLS emulates the broadcast domain of a LAN across an MPLS network cloud. Traditional MPLS implementations of VPLS require that all participating ingress PE routers make separate copies of each broadcast or multicast packet to send to all other PE routers that are part of the VPLS site for the same extended LAN. In a large virtual private network (VPN), replication overhead can be significant for each ingress router and its attached core-facing links.

Junos OS offers the following VPLS enhancements which provide redundancy for VPLS between PE and CE routers:

- Redundancy using BGP for multihomed links between PE and CE devices— Juniper Networks integrates the local preference and path selection capability of BGP with VPLS to allow a CE Ethernet switch to have a backup path across the network.

- Redundancy using the Spanning Tree Protocol (STP) for multihomed links between PE and CE devices — Various versions of STP can be used in the CE network to avoid loops in a multihoming environment. The provider does not have any control over this customer network configuration. The provider can also implement BGP-based loop avoidance as an additional measure to avoid loops.

The following standardized VPLS implementations are supported by the Internet Engineering Task Force (IETF):

- RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*

- RFC 4762, *Virtual Private LAN Service (VPLS) Using LDP Signaling*

For more information about the basic configuration of next-generation VPLS, see the Technology Overview *Next-Generation VPLS Using Point-to-Multipoint LSPs for Unicast and Multicast Forwarding*.

For a detailed technology overview of VPLS, you can refer to *LDP-BGP VPLS Interworking* at the following location: https://www.juniper.net/us/en/local/pdf/whitepapers/2000282-en.pdf .

Redundancy Using BGP for Multihomed Links between PE and CE Routers

Juniper Networks implements a BGP-based multihoming solution to provide redundancy for VPLS between PE and CE routers.

In this implementation:

- VPLS-enabled PE routers (also called VPLS PE routers) collectively elect one of the VPLS PE routers, to which a site is multihomed, as the designated forwarder of traffic between this site and all other sites.
- All the other VPLS PE routers, to which the same site is connected, do not forward traffic to or from the site.
- Essentially all VPLS PE routers behave as if the site is singlehomed to the VPLS PE router that is the designated forwarder.
- Service providers are able to prevent well-known Layer 2 loops without relying on the customer’s STP configuration.
- Customers can still run STP as a fallback strategy to prevent loops that are formed without the service provider's knowledge.

The benefits of multihoming include:

- Redundancy of the link connecting the PE router and the CE device.
- Redundancy of the directly connected PE routers.
- Faster convergence when there is a link failure between a PE router and CE device.
- The same BGP attributes are used to configure primary and backup links.

Operation of Next-Generation VPLS for Multicast with Multihoming Using BGP

VPLS provides a multipoint-to-multipoint Ethernet service that can span one or more metro areas and multiple sites. VPLS provides connectivity as if these sites are attached to the same Ethernet LAN.

VPLS uses an IP and MPLS service provider infrastructure. From the service provider's point of view, using IP and MPLS routing protocols and procedures instead of STP, and using MPLS labels instead of VLAN identifiers (IDs), significantly improves the scalability of the VPLS service.

Single CE Site Connected to Multiple VPLS PE Routers

This section describes the process used to elect a single designated forwarder for a multihomed site.

For a multihomed site, all the PE routers in the VPLS instance elect the same designated forwarder PE router using the BGP VPLS multihoming procedure. Only elected designated forwarders forward traffic to and receive traffic from the multihomed site. All other PE routers where this multihomed site is present do not participate in forwarding for that site.

All remote PE routers are aware of the designated forwarder PE router for each multihomed site and do not create a pseudowire to the PE routers that are not the designated forwarder for the multihomed site.

In Figure 72 on page 997:

- The same site ID (sometimes known as a VPLS edge identifier or VE ID) is configured on all VPLS PE routers to which a site is multihomed.
- All PE routers are aware of which sites are multihomed since they see multiple advertisements with the same site ID.
- One of the VPLS PE routers is selected as the designated forwarder for this site by all PE routers based on a deterministic algorithm.
- The algorithm selects the VPLS PE router that originates the best advertisement with a particular site ID as the designated forwarder. There are two possible selection methods:
  - BGP path selection on the route reflector and the PE routers
  - VPLS site selection on the PE router only
- If multiple network layer reachability information (NLRI) advertisements have the same route distinguisher and site ID, the router uses BGP path selection rules to select the best path. The BGP rules are:
  - Always prefer advertisements that do not have the down bit set over ones that do have this bit set.
  - Prefer the advertisement with the higher local preference.
  - Use the configurable per-site site preference to set the BGP local preference in the advertisement and influence the choice of the designated forwarder.
  - Ignore the interior gateway protocol (IGP) metric while doing path selection because the choice of designated forwarder must be the same on all PE routers.

- Among advertisements with the same route distinguisher, apply VPLS site selection rules (a subset of BGP path selection rules) to pick the select advertisement.

Figure 72 on page 997 illustrates the following four-step process to select the designated forwarder and create the pseudowire:

1. Router PE1 and Router PE2 both have the same site ID (Site 1) for Router CE1.

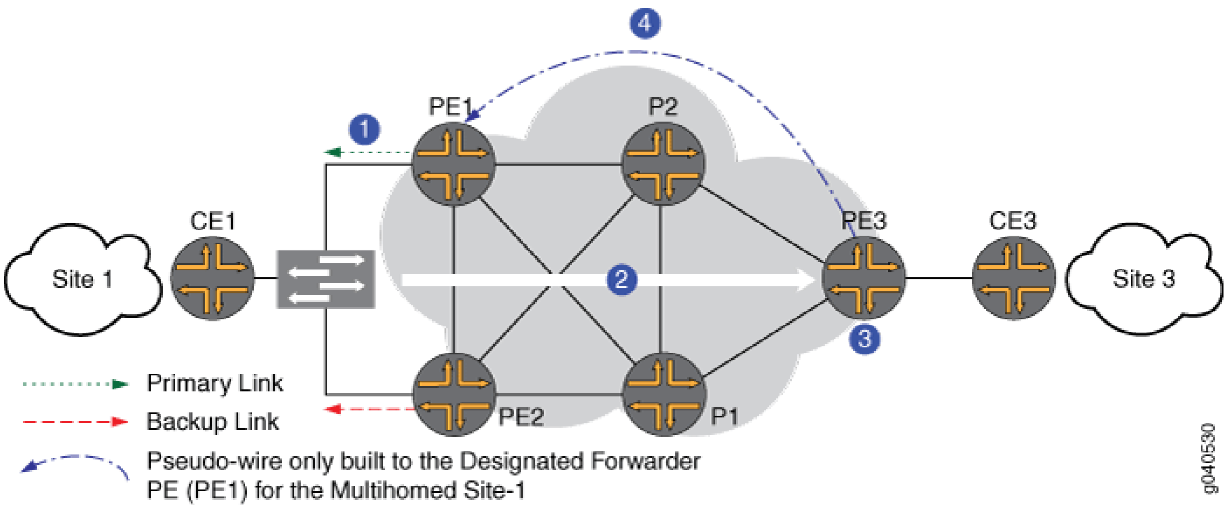
   Router PE1 has a better local preference of 65535 and is configured as the primary router.

2. Router PE3 receives the BGP NLRI advertisement from Router PE1 and Router PE2 with the local preferences of 65535 and 1, respectively.

3. Router PE3 runs the BGP path selection algorithm and selects Router PE1 as the designated forwarder VPLS edge PE router for Site 1.

4. Router PE3 creates the pseudowire only to Router PE1, which helps to save bandwidth in the network core.

Figure 72: Single CE Site Multihomed with Two PE Routers


4
PE1
P2
1
CE1
Site 1
2
PE3
CE3
Site 3
3
PE2
P1
Primary Link
Backup Link
Pseudo-wire only built to the Designated Forwarder PE (PE1) for the Multihomed Site-1
g040530

The resulting VPLS PE router roles for Site 1 are:

- Router PE1 is the designated forwarder VPLS edge PE router.

- Router PE2 is the non-designated forwarder VPLS edge PE router.

- Router PE3 is the remote VPLS edge PE router.

All the interfaces linking the CE and PE devices that are connected to the designated forwarder VPLS PE router, are marked `Up` and `forwarding` in `show` command output.

All the interfaces linking the CE and PE devices on the non-designated forwarder VPLS PE router, are marked `vc-down` in `show` command output. The router does not send traffic or forward received traffic on these interfaces.

Remote VPLS PE routers establish pseudowires only to the designated PE router, and tear down any pseudowires to the non-designated PE router.

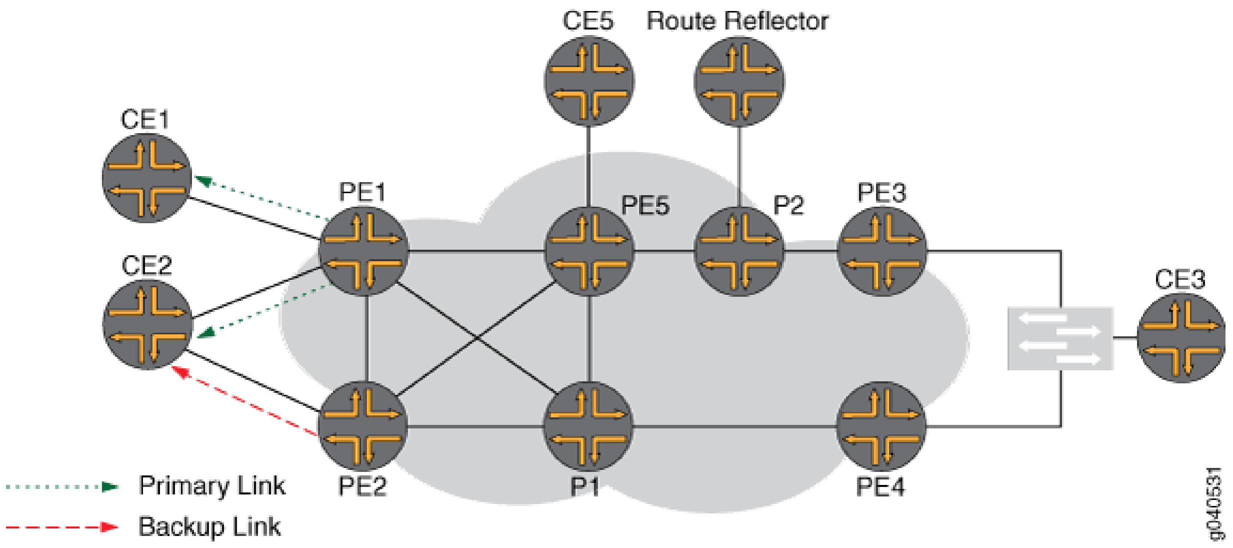
Multiple CE Sites Connected to a Single VPLS PE Router for Link Redundancy

This section describe some of the operational details of multiple CE sites connected to a single VPLS PE router.

In Figure 73 on page 998:

- Router CE2 is multihomed to Router PE1 and Router PE2.
- Router CE1 is singlehomed to Router PE1.

Figure 73: Two CE Sites Multihomed to a Single PE Router on Different Line Cards



The scenario shown in Figure 2 is common. Your network might have a single PE router in a remote area, but you would like to multihome a Layer 2 network to different Flexible PIC Concentrators (FPCs) on the same PE router. This configuration provides link redundancy on the CE devices and link redundancy on the links between the CE and PE devices, but limited link redundancy on PE devices. In this case, you need the ability to configure a site to use a single active interface for forwarding.

In this scenario:

- Path selection is done per site to determine if a PE router is the designated forwarder for that site or not.

- Only a single pseudowire is established between any two PE routers, even if one or both of them have multiple designated PE routers.

> **NOTE**: A pseudowire between two PE routers is always established between the designated sites with the minimum site IDs on the two PE routers.

- Establishing a single pseudowire avoids the need to maintain multiple flooding and media access control (MAC) address tables per instance (one per site) on each PE router.
- The local interfaces are marked `vc-down` in the `show` command output where a site is connected to the non-designated forwarder router.
- When a designated site on a PE router fails, all MAC addresses from this remote PE router have to be learned again, since the router does not know the exact site where the MAC addresses were originally learned from.

Implementation of Redundancy Using VPLS Multihomed Links Between PE and CE Devices

You might need to multihome a CE device to multiple PE routers without causing a Layer 2 forwarding loop. This is not a problem if the CE device is a router, since no Layer 2 loops can form when using a router. However, if the CE device is a Layer 2 device, like a hub or switch, multihoming it to two PE routers can cause a Layer 2 loop.

You can use one of the following methods to prevent the Layer 2 loop:

- BGP-based primary and backup link selection.
- Spanning tree protocol (STP) to prune links to the CE router. However, this method requires the service provider to trust its customer to not cause any Layer 2 loops by misconfiguration.
- Active and standby up link functionality, such as the redundant trunk groups that are supported on Juniper Networks EX Series Ethernet Switches.

The limitations of using STP on the CE site are:

- Backbone and access network bandwidth is not used efficiently.
- PE routers using STP to prevent loops with dual-homed sites receive broadcast traffic unnecessarily because the pseudowire to the standby PE router still exists.
- When the direct link between the CE and PE router fails, multihoming works fine. When a link connected downstream from the CE router fails, multihoming does not work.

The benefits and properties of the BGP-based solution are:

- BGP path selection does not have the limitations of STP.
- A CE device that is multihomed to multiple PE routers is given the same site ID on all the PE routers it is multihomed to.
- The BGP path selection algorithm selects the router that originates the best advertisement as the VPLS PE designated forwarder.
- If desired, you can set the local preference on the PE routers to control BGP path selection.
- BGP path selection occurs on the route reflector and the PE router.
- An IGP metric is not part of the selection process.
- If the route distinguisher is the same on both PE routers, the route reflector selects one PE router as the designated forwarder. If the route distinguishers are different on the PE routers, the route reflector forwards both copies of the route to the remote PE routers.

RELATED DOCUMENTATION

Example: Next-Generation VPLS for Multicast with Multihoming | **1000**

Example: NG-VPLS Using Point-to-Multipoint LSPs | **1034**

Next-Generation VPLS Point-to-Multipoint Forwarding Overview | **1028**

Example: Next-Generation VPLS for Multicast with Multihoming

IN THIS SECTION

- Requirements | **1000**
- Overview and Topology | **1001**
- Configuration | **1004**

This example shows how to configure next-generation VPLS for multicast with multihoming. It is organized in the following sections:

Requirements

The following table lists the hardware and software requirements for this configuration.

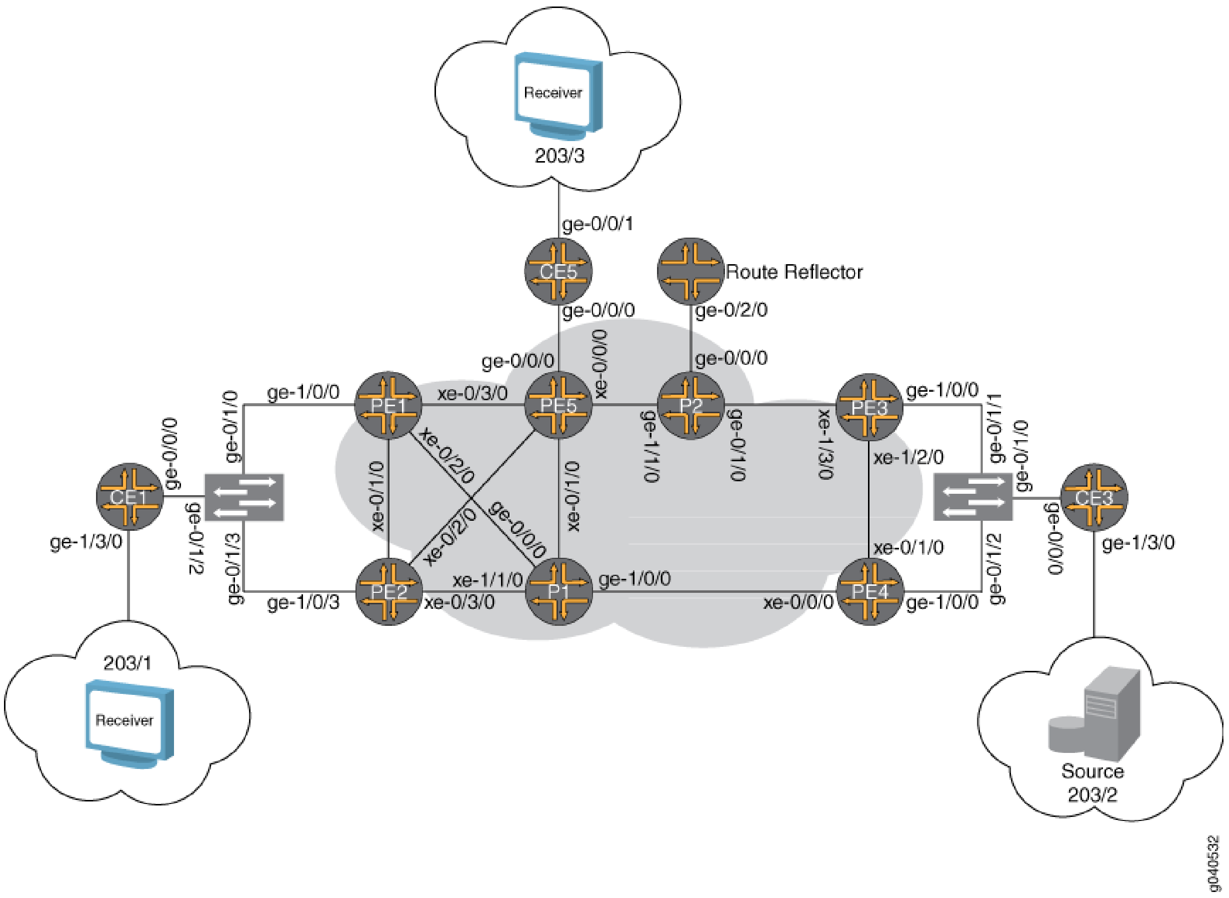
Table 23: Hardware and Software Used

Equipment	Components	Software
Four MX Series 5G Universal Routing Platforms	DPC40X-1GE -X, DPC 4X-10GE-X, DPC40x-1GE-R, DPC 4X-10GE-R	Junos OS Release 9.3 or later
Two M320 Multiservice Edge Routers and T Series Core Routers	FPC 3, 10GE Xenpak	Junos OS Release 9.3 or later
Five EX Series Ethernet Switches	EX4200, EX3200	Junos OS Release 9.4 or later

Overview and Topology

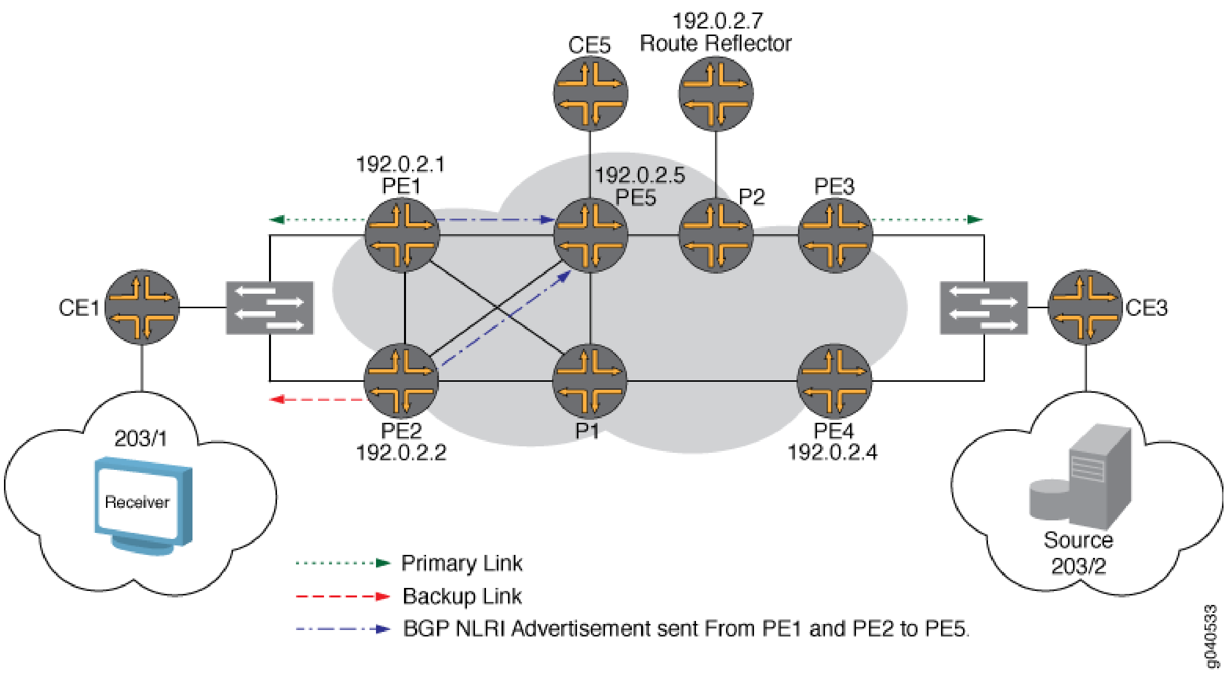
Figure 74 on page 1002 shows the physical topology used in this next-generation VPLS multihoming example.

Figure 74: Physical Topology of Next-Generation VPLS for Multicast with Multihoming



[Figure 75 on page 1003](#) show the logical topology of the next-generation VPLS multihoming example.

Figure 75: Logical Topology of Next-Generation VPLS for Multicast with Multihoming



The network state and configuration before the implementation is as follows:

- Five PE routers participating in the next-generation VPLS domain named GOLD.
- OSPF, BGP, and RSVP are configured on the MPLS core interfaces.
- The `no-tunnel-services` statement is included in the VPLS routing instance. This statement supports the use of label-switched interface (LSI) tunnel interfaces for VPLS.
- Router PE1 and Router PE2 are configured with a dynamic point-to-multipoint LSP using the `vpls-GOLD-p2mp-template` template.
- Router PE3 and Router PE4 are configured to use static point-to-multipoint LSPs.

> **NOTE**: Single-hop point-to-multipoint LSPs are not supported, so single-hop point-to-multipoint LSPs are down.

- Router CE1 is multihomed to Router PE1 and Router PE2 through an EX4200 Layer 2 switch.
- Router CE3 is multihomed to Router PE3 and Router PE4 through an EX4200 Layer 2 switch.
- Router CE5 is singlehomed to Router PE5.
- The off-path route reflector is configured for BGP. The `family l2vpn` statement is included in the route reflector configuration.

- Router CE3 is connected to test equipment through port 203/2. The test equipment generates multicast traffic to groups 203.0.113.1 through 203.0.113.10 at the rate of 10,000 pps.
- Router CE1 and Router CE5 are configured with static Internet Group Management Protocol (IGMP) joins so they can receive the multicast traffic from Router CE3.
- The Layer 2 switches are configured with truck ports to the PE routers and access ports to the test equipment.

Here is a summary of the steps necessary to complete the configuration successfully:

1. Configure a unique route distinguisher for the VPLS routing instance named GOLD on Router PE1, Router PE2, Router PE3, and Router PE4.
2. Configure the same site ID for the multihomed PE routers. Configure both Router PE1 and Router PE2 with a site ID value of 1. Configure both Router PE3 and Router PE4 with a site ID value of 3.
3. Configure multihoming under the CE1 site configuration.
4. Configure the site-preference `Primary` on Router PE1 and configure the site-preference `Backup` on Router PE2. In this case, Router PE1 has the primary link to Router CE1 and Router PE2 has the backup link to Router CE1.
5. Configure the site preference on Router PE3 and Router PE4. Configure Router PE3 as the primary and Router PE4 as the backup.

Configuration

IN THIS SECTION

- Configuring Next-Generation VPLS Multihoming | **1005**
- Validating the VPLS Control Plane | **1006**
- Verifying the VPLS Data Plane | **1014**
- Results | **1019**

This section provides a step-by-step procedure to configure next-generation VPLS for multicast with multihoming.

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the `commit check` command.

This example is organized in the following sections:

Configuring Next-Generation VPLS Multihoming

Step-by-Step Procedure

1. In BGP-based VPLS multihoming, it is recommended that you configure distinct route distinguishers for each multihomed router. Configuring distinct route distinguishers helps with faster convergence when the connection to a primary router goes down. It also requires the other backup PE routers to maintain additional state information for faster convergence.

   There are two levels of path selection:

   - The first is BGP: BGP uses a combination of route distinguisher, site ID, and VE block offset for BGP path selection.
   - The second is in VPLS: VPLS uses the site ID for VPLS path selection.

   By configuring unique route distinguishers, the prefixes for BGP path selection are all unique. Therefore, BGP path selection is skipped and VPLS path selection is used, which only looks at the site ID.

   On Router PE1, Router PE2, Router PE3, and Router PE4 configure a unique router distinguisher for the `GOLD` routing instance.

   ```
   user@PE1# set routing-instance GOLD route-distinguisher 192.0.2.1:1
   user@PE2# set routing-instance GOLD route-distinguisher 192.0.2.2:10
   user@PE3# set routing-instance GOLD route-distinguisher 192.0.2.3:1
   user@PE4# set routing-instance GOLD route-distinguisher 192.0.2.4:10
   ```

2. Configure site ID `1` on Routers PE1 and PE2 for Router CE1. Configure site ID `3` on Routers PE3 and PE4 for Router CE3.

   ```
   user@PE1# set routing-instance GOLD protocols vpls site CE1 site-identifier 1
   user@PE2# set routing-instance GOLD protocols vpls site CE1 site-identifier 1
   user@PE3# set routing-instance GOLD protocols vpls site CE3 site-identifier 3
   user@PE4# set routing-instance GOLD protocols vpls site CE3 site-identifier 3
   ```

3. Enable multihoming by including the `multi-homing` statement under the multihomed site configuration on Router PE1, Router PE2, Router PE3, and Router PE4.

```
user@PE1# set routing-instance GOLD protocols vpls site CE1 multi-homing
user@PE2# set routing-instance GOLD protocols vpls site CE1 multi-homing
user@PE3# set routing-instance GOLD protocols vpls site CE3 multi-homing
user@PE4# set routing-instance GOLD protocols vpls site CE3 multi-homing
```

4. Include the `site-preference primary` statement on Router PE1 and Router PE3, and include the `site-preference backup` statement on Router PE2 and Router PE4. The `site-preference primary` statement sets the local preference to the highest value (65535) and the `site-preference backup` statement sets the BGP local preference to 1. Since the site ID is the same, the routers select the highest local preference value as the designated forwarder.

```
user@PE1# set routing-instance GOLD protocols vpls site CE1 site-preference primary
user@PE2# set routing-instance GOLD protocols vpls site CE1 site-preference backup
user@PE3# set routing-instance GOLD protocols vpls site CE3 site-preference primary
user@PE4# set routing-instance GOLD protocols vpls site CE3 site-preference backup
```

Validating the VPLS Control Plane

Step-by-Step Procedure

This section presents show commands that you can use to verify the operation of the example configuration.

In this example the traffic patterns are:

- The source is connected to Router CE3 and sends 10,000 pps for the groups 203.0.113.1 to 203.0.113.10. Router CE3 is configured as a rendezvous point.

- Multicast receivers are connected to both Router CE1 and Router CE5. Protocol Independent Multicast (PIM) join messages are generated by the test equipment.
- The link between Router PE3 and Router CE3 and the link between Router PE1 and Router CE1 are configured as primaries for VPLS multihoming.
- All PE routers have a BGP session with the route reflector.
- All PE routers have a label-switched path (LSP) that is created to the route reflector so that the PE routers have a route to the route reflector in the `inet.3` table for route resolution.

1. On Router PE1, use the `show vpls connections` command to verify that the VPLS connections are `Up` between Router PE1 and Router PE3 and between Router PE1 and PE5. Router PE1 is the primary link selected by the VPLS multihoming configuration.

```
user@PE1# show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection

Legend for interface status
Up -- operational
Dn -- down

Instance: GOLD
  Local site: CE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   RN
    3                         rmt   Up     Nov 16 11:22:44 2009           1
```

```
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 262147, Outgoing label: 262145
      Local interface: lsi.1048835, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 3
    5                         rmt   Up     Nov 16 11:22:46 2009           1
      Remote PE: 192.0.2.5, Negotiated control-word: No
      Incoming label: 262149, Outgoing label: 262161
      Local interface: lsi.1048836, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 5
```

2. On Router PE2, use the `show vpls connections` command to verify that the VPLS connections to Router PE3 and Router PE5 are in the `LN` state, meaning the local router is not the designated forwarder. Router PE2 is configured to be the backup link for Router CE1.

```
user@PE2# show vpls connections
...

Instance: GOLD
  Local site: CE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   LN
    3                         rmt   LN
    5                         rmt   LN
```

3. On Router PE3, use the `show vpls connections` command to verify that the VPLS connections to Router PE1 and Router PE5 are `Up`. Router PE3 is configured to be the primary link for Router CE3.

```
user@PE3# show vpls connections
...

Instance: GOLD
  Local site: CE3 (3)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Nov 16 11:22:01 2009           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 262145, Outgoing label: 262147
      Local interface: lsi.1048832, Status: Up, Encapsulation: VPLS
```

```
          Description: Intf - vpls GOLD local site 3 remote site 1
      3                         rmt   RN
      5                         rmt   Up     Nov 16 11:22:56 2009           1
        Remote PE: 192.0.2.5, Negotiated control-word: No
        Incoming label: 262149, Outgoing label: 262163
        Local interface: lsi.1048834, Status: Up, Encapsulation: VPLS
          Description: Intf - vpls GOLD local site 3 remote site 5
```

4. On Router PE4, use the `show vpls connections` command to verify that the VPLS connections are in the `LN` state, meaning the local site is not designated. Router PE4 is configured to be the backup link for Router CE3.

```
user@PE4# show vpls connections
...

Instance: GOLD
  Local site: CE3 (3)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   LN
    3                         rmt   SC
    5                         rmt   LN
```

5. On Router PE1, use the `show route advertising-protocol` command to verify that Router PE1 (the multihoming primary router) is sending the BGP Layer 2 VPN route advertisement to the route reflector with the local preference value of `65535`. The local preference is used by Router PE3 to select Router PE1 as the designated forwarder, rather than selecting Router PE2 that has a local preference of `1`.

```
user@PE1# show route advertising-protocol bgp 192.0.2.7 extensive
GOLD.l2vpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 192.0.2.1:1:1:1/96 (1 entry, 1 announced)
 BGP group to-RR type Internal
     Route Distinguisher: 192.0.2.1:1
     Label-base: 262145, range: 8
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 65535
     AS path: [65000] I
     Communities: target:65000:1 Layer2-info: encaps:VPLS, control flags:, mtu: 0, site
```

```
preference: 65535
     PMSI: Flags 0:RSVP-TE:label[0:0:0]:Session_13[192.0.2.1:0:9519:192.0.2.1]
```

6. On Router PE2, use the `show route advertising-protocol` command to verify that Router PE2 is configured as the multihoming backup with a local preference of `1`.

```
user@PE2# show route advertising-protocol bgp 192.0.2.7 extensive
GOLD.l2vpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 192.0.2.2:10:1:1/96 (1 entry, 1 announced)
 BGP group to-RR type Internal
     Route Distinguisher: 192.0.2.2:10
     Label-base: 262145, range: 8
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 1
     AS path: [65000] I
     Communities: target:65000:1 Layer2-info: encaps:VPLS, control flags:, mtu: 0, site
preference: 1
```

7. On Router PE3, use the `show route receive-protocol` command to verify that Router PE3 receives the Layer 2 VPN route from the route reflector for Router PE1 and Router PE2 with different local preference values.

   BGP route selection is based on the received `l2vpn` routes for the VPLS site connected to multihomed PE routers. Since the route distinguishers are different on Router PE1 and Router PE2, Router PE3 and Router PE4 consider the received routes from Router PE1 and Router PE2 as different routes. Router PE3 and Router PE4 run the BGP path selection algorithm and select Router PE1, the router advertising the route with the higher local preference value, as the designated forwarder.

```
user@PE3# show route receive-protocol bgp 192.0.2.7
bgp.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  192.0.2.1:1:1:1/96
*                         192.0.2.1                      65535      I
  192.0.2.2:10:1:1/96
*                         192.0.2.2                      1          I
  192.0.2.4:10:3:1/96
*                         192.0.2.4                      1          I
  192.0.2.5:10:5:1/96
*                         192.0.2.5                      100        I
```

```
GOLD.l2vpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop             MED     Lclpref    AS path
  192.0.2.1:1:1:1/96
*                         192.0.2.1


                        65535      I
  192.0.2.2:10:1:1/96
*                         192.0.2.2


                        1          I
  192.0.2.4:10:3:1/96
*                         192.0.2.4                     1          I
  192.0.2.5:10:5:1/96
*                         192.0.2.5                     100        I
```

8. On Router PE3, use the `show route table` command to verify that Router PE3 has selected the static point-to-multipoint LSP from Router PE3 to Router PE1 for forwarding.

   Notice that Router PE2 does not have any provider multicast service interface (PMSI) flags because PMSI attributes are not attached.

```
user@PE3# show route table GOLD.l2vpn.0 extensive
GOLD.l2vpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
192.0.2.1:1:1:1/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-65536
                Route Distinguisher: 192.0.2.1:1
                PMSI: Flags 0:RSVP-TE:label[0:0:0]:Session_13[192.0.2.1:0:9519:192.0.2.1]
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 192.0.2.7
                Protocol next hop: 192.0.2.1
                Indirect next hop: 2 no-forward
                State: <Secondary Active Int Ext>
                Local AS: 65000 Peer AS: 65000
                Age: 2:30:44    Metric2: 1
                Task: BGP_65000.192.0.2.7+179
                Announcement bits (1): 0-GOLD-l2vpn
                AS path: I (Originator) Cluster list:  192.0.2.7
                AS path:  Originator ID: 192.0.2.1
                Communities: target:65000:1 Layer2-info: encaps:VPLS, control flags:, mtu: 0,
site preference: 65535
```

```
                Import Accepted
                Label-base: 262145, range: 8
                Localpref: 65535
                Router ID: 192.0.2.7
                Primary Routing Table bgp.l2vpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.0.2.1 Metric: 3
                        Indirect next hop: 2 no-forward
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 10.10.8.2 via xe-0/1/0.0 weight 0x1
                        192.0.2.1/32 Originating RIB: inet.3
                          Metric: 3                       Node path count: 1
                          Forwarding nexthops: 1
                                Nexthop: 10.10.8.2 via xe-0/1/0.0

192.0.2.2:10:1:1/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-2
                Route Distinguisher: 192.0.2.2:10
                Next hop type: Indirect
                Next-hop reference count: 3
                Source: 192.0.2.7
                Protocol next hop: 192.0.2.2
                Indirect next hop: 2 no-forward
                State: <Secondary Active Int Ext>
                Local AS: 65000 Peer AS: 65000
                Age: 2:30:44    Metric2: 1
                Task: BGP_65000.192.0.2.7+179
                Announcement bits (1): 0-GOLD-l2vpn
                AS path: I (Originator) Cluster list:  192.0.2.7
                AS path:  Originator ID: 192.0.2.2
                Communities: target:65000:1 Layer2-info: encaps:VPLS, control flags:, mtu: 0,
site preference: 1
                Import Accepted
                Label-base: 262145, range: 8
                Localpref: 1
                Router ID: 192.0.2.7
                Primary Routing Table bgp.l2vpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.0.2.2 Metric: 3
                        Indirect next hop: 2 no-forward
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
```

```
                                  Next hop: 10.10.8.2 via xe-0/1/0.0 weight 0x1
                          192.0.2.2/32 Originating RIB: inet.3
                            Metric: 3                       Node path count: 1
                            Forwarding nexthops: 1
                                  Nexthop: 10.10.8.2 via xe-0/1/0.0
```

9. On Router PE3, use the `show vpls connections` command to verify that the VPLS connection is in the `Up` state.

   Notice the display also shows the local interface and the incoming and outgoing label values used.

```
user@PE3# show vpls connections extensive
...

Instance: GOLD
  Local site: CE3 (3)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    IRB interface present: no
    ge-1/0/0.1
    lsi.1048832        1         Intf - vpls GOLD local site 3 remote site 1
    lsi.1048833        2         Intf - vpls GOLD local site 3 remote site 2
        Interface flags: VC-Down
    lsi.1048834        5         Intf - vpls GOLD local site 3 remote site 5
        Interface flags: VC-Down
    Label-base        Offset     Range     Preference
    262145            1          8         65535
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Nov 16 11:22:01 2009           1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 262145, Outgoing label: 262147
      Local interface: lsi.1048832, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 3 remote site 1
      RSVP-TE P2MP lsp:
        Egress branch LSP:  192.0.2.3:192.0.2.1:1:vpls:GOLD, State: Up
    Connection History:
        Nov 16 11:22:54 2009  PE route changed
        Nov 16 11:22:01 2009  status update timer
        Nov 16 11:22:01 2009  PE route changed
        Nov 16 11:22:01 2009  Out lbl Update                    262147
        Nov 16 11:22:01 2009  In lbl Update                     262145
        Nov 16 11:22:01 2009  loc intf up                  lsi.1048832
    3                         rmt   RN
    5                         rmt   RD
  Ingress RSVP-TE P2MP LSP: vpls-GOLD, Flood next-hop ID: 616
```

Verifying the VPLS Data Plane

Step-by-Step Procedure

After the control plane is verified using the previous steps, you can verify the data plane. The data plane operation in the VPLS multihoming scenario is the same as the regular next-generation VPLS operation. This section describes the `show` command outputs that you can use to validate the data plane.

1. On Router PE3, use the `show mpls lsp` command to verify the state of the static LSPs and sub-LSPs.

   Router PE2 is configured with static point-to-multipoint LSPs and sub-LSPs with link protection. Point to multipoint LSPs are not supported for single-hop LSPs. In the following output notice that the single-hop point-to-multipoint LSP from Router PE3 to Router PE4 is `down`.

```
user@PE3# show mpls lsp p2mp ingress
Ingress LSP: 1 sessions
P2MP name: vpls-GOLD, P2MP branch count: 4
To              From            State Rt P     ActivePath       LSPname
192.0.2.5         192.0.2.3        Up     0 *                       to-pe5
192.0.2.1         192.0.2.3        Up     0 *                       to-pe1
192.0.2.4         192.0.2.3        Dn     0 *                       to-pe4
192.0.2.2         192.0.2.3        Up     0 *                       to-pe2
Total 4 displayed, Up 3, Down 1
```

2. On Router PE1, use the `show mpls lsp` command to verify the state of the dynamic LSPs.

   Router PE1 is using a dynamic point-to-multipoint LSP template configured with link protection. Notice that the LSP state is `Up` and that link protection is `desired`.

```
user@PE1# show mpls lsp p2mp ingress extensive
Ingress LSP: 1 sessions
P2MP name: 192.0.2.1:1:vpls:GOLD, P2MP branch count: 1

192.0.2.3
  From: 192.0.2.1, State: Up, ActiveRoute: 0, LSPname: 192.0.2.3:192.0.2.1:1:vpls:GOLD
  ActivePath:  (primary)
  P2MP name: 192.0.2.1:1:vpls:GOLD
  Link protection desired
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 45 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 3)
 10.10.3.2 S 10.10.9.2 S 10.10.8.1 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt):
          10.10.3.2(Label=488645) 192.0.2.4(flag=0x21) 10.10.9.2(flag=1 Label=299936)
10.10.8.1(Label=262145)
   12 Nov 16 15:38:08.116 CSPF: computation result ignored[314 times]
   11 Nov 16 11:23:44.856 Link-protection Up
   10 Nov 16 11:23:32.696 CSPF: computation result ignored[3 times]
    9 Nov 16 11:22:47.859 Record Route:  10.10.3.2(Label=488645) 192.0.2.4(flag=0x21)
10.10.9.2(flag=1 Label=299936) 10.10.8.1(Label=262145)
    8 Nov 16 11:22:44.910 Record Route:  10.10.3.2(Label=488645) 192.0.2.4(flag=0x20)
10.10.9.2(Label=299936) 10.10.8.1(Label=262145)
    7 Nov 16 11:22:44.910 Up
    6 Nov 16 11:22:44.910 10.10.3.1: Down
    5 Nov 16 11:22:44.866 Selected as active path
    4 Nov 16 11:22:44.864 Record Route:  10.10.3.2(Label=488629) 192.0.2.4(flag=0x20)
10.10.9.2(Label=299920) 10.10.8.1(Label=3)
    3 Nov 16 11:22:44.864 Up
    2 Nov 16 11:22:44.852 Originate Call
    1 Nov 16 11:22:44.852 CSPF: computation result accepted  10.10.3.2 10.10.9.2 10.10.8.1
  Created: Mon Nov 16 11:22:45 2009
Total 1 displayed, Up 1, Down 0
```

3. On Router PE3, use the `monitor interface traffic` command to verify the multicast replication behavior for the point-to-multipoint LSP on the designated forwarder Router PE3.

   The output shows that `10,000` pps are received on interface `ge-1/0/0` from Router CE3. The traffic has been forwarded to the provider (P) Router P2 and Router PE4 through xe-0/0/0 and xe-0/1/0, respectively. Based on the output, you can determine that a single copy of the packet is being sent to Router P2 and Router PE4.

```
user@PE3> monitor interface traffic
PE3                        Seconds: 8                   Time: 11:58:40
```

```
Interface    Link  Input packets        (pps)     Output packets        (pps)
 lc-0/0/0      Up              0                               0
 xe-0/0/0      Up       13570505          (0)         4507338866      (10000)
 lc-0/1/0      Up              0                               0
 xe-0/1/0      Up         292843          (1)          628972219      (10000)
 lc-0/2/0      Up              0                               0
 xe-0/2/0      Up         343292          (0)             206808          (1)
 lc-0/3/0      Up              0                               0
 xe-0/3/0    Down              0          (0)                  0          (0)
 ge-1/0/0      Up     2703709733       (9999)           13203544          (1)
 lc-1/0/0      Up              0                               0
 ge-1/0/1    Down    50380341937          (0)        60024542111          (0)
 ge-1/0/2    Down    60652323068          (0)        84480825838          (0)
 ge-1/0/3    Down    81219536264          (0)        84614255165          (0)
 ge-1/0/4    Down    54379241112          (0)        83656815208          (0)
```

4. On Router P2, use the `monitor interface traffic` command to verify that the multicast packet replication happens close to the PE routers connected to the receivers.

   Router PE1 and Router PE5 are connected to receivers that have joined this multicast group. Notice that incoming multicast packets from Router PE3 on the `ge-0/1/0` interface are replicated twice and sent out on the `ge-1/1/0` interface.

```
user@P2> monitor interface traffic
P2                        Seconds: 6                   Time: 12:07:58

Interface    Link  Input packets        (pps)     Output packets        (pps)
 ge-0/1/0      Up      661459806      (10000)             116236          (0)
 ge-1/1/0      Up         115956          (0)         1322690473      (20000)
 gr-2/1/0      Up              0          (0)                  0          (0)
 ip-2/1/0      Up              0          (0)                  0          (0)
```

5. On Router PE3, use the `show vpls flood` command to verify information about the flood next-hop route.

Junos OS Release 9.0 and later identifies the flood next-hop route as a composite next hop. Notice that the interface is `ge-1/0/0.1`, the next-hop type is `composite`, and that the flood composition is `flood-to-all`. This means the traffic is flooded to all the PE routers.

```
user@PE3# show vpls flood extensive
Name: GOLD
CEs: 1
VEs: 1
  Flood route prefix: 0x30002/51
  Flood route type: FLOOD_GRP_COMP_NH
  Flood route owner: __ves__
  Flood group name: __ves__
  Flood group index: 0
  Nexthop type: comp
  Nexthop index: 606
    Flooding to:
    Name             Type          NhType          Index
    __all_ces__      Group          comp             603
        Composition: split-horizon
        Flooding to:
        Name             Type          NhType          Index
        ge-1/0/0.1       CE             ucst             578

  Flood route prefix: 0x30003/51
  Flood route type: FLOOD_GRP_COMP_NH
  Flood route owner: __all_ces__
  Flood group name: __all_ces__
  Flood group index: 1
  Nexthop type: comp
  Nexthop index: 611
    Flooding to:
    Name             Type          NhType          Index
    __ves__          Group          comp             594
        Composition: flood-to-all
    Component p2mp NH (for all core facing interfaces):
    Index
    616
    Flooding to:
    Name             Type          NhType          Index
    __all_ces__      Group          comp             603
        Composition: split-horizon
        Flooding to:
        Name             Type          NhType          Index
        ge-1/0/0.1       CE             ucst            578

  Flood route prefix: 0x30001/51
  Flood route type: FLOOD_GRP_COMP_NH
  Flood route owner: __re_flood__
  Flood group name: __re_flood__
  Flood group index: 65534
  Nexthop type: comp
  Nexthop index: 598
    Flooding to:
    Name             Type          NhType          Index
    __ves__          Group          comp            594
        Composition: flood-to-all
    Component p2mp NH (for all core facing interfaces):
    Index
    616
    Flooding to:
    Name             Type          NhType          Index
    __all_ces__      Group          comp            603
        Composition: split-horizon
        Flooding to:
        Name             Type          NhType          Index
        ge-1/0/0.1       CE             ucst            578
Name: __juniper_private1__
CEs: 0
VEs: 0
```

6. On Router PE3, use the `show vpls mac-table` command to verify that the MAC address of the PE router at the remote end of the VPLS has been learned and added to the MAC address table.

   Notice that the MAC address is learned on the `ge-1/0/0.1` interface.

```
user@PE3# show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC,
           SE -Statistics enabled, NM -Non configured MAC)

Routing instance : GOLD
 Bridging domain : __GOLD__, VLAN : NA
   MAC                 MAC      Logical
   address             flags    interface
   00:14:f6:75:78:00   D        ge-1/0/0.1
```

7. On Router PE3, use the `show route forwarding-table` command to verify that the forwarding table has the required entries with two labels: one for the VPLS service and the other for the next-hop interface.

```
user@PE3> show route forwarding-table family vpls vpn GOLD
Routing table: GOLD.vpls
VPLS:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    dscd   574     1
lsi.1048832        intf     0                    indr 1048575     4
                              10.10.7.1         Push 262147, Push 309680(top)   596     2
xe-0/0/0.0
lsi.1048836        intf     0                    indr 1048574     4
                              10.10.7.1         Push 262179, Push 299856(top)   589     2
xe-0/0/0.0
00:10:db:e9:4e:b6/48
                   user     0                    indr 1048574     4
                              10.10.7.1         Push 262179, Push 299856(top)   589     2
xe-0/0/0.0
00:12:1e:c6:98:00/48
                   user     0                    indr 1048575     4
                              10.10.7.1         Push 262147, Push 309680(top)   596     2
xe-0/0/0.0
00:14:f6:75:78:00/48
                   user     0                    ucst   578     4 ge-1/0/0.1
0x30002/51         user     0                    comp   606     2
ge-1/0/0.1         intf     0                    ucst   578     4 ge-1/0/0.1
0x30003/51         user     0                    comp   611     2
0x30001/51         user     0                    comp   598     2
```

Results

The configuration and verification parts of this example have been completed. The following section is for your reference.

The relevant sample configuration for Router PE1 follows:

Router PE1

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ethernet;
}
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.1/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.3.1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/0 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface xe-0/3/0.0 {
            link-protection;
        }
        interface xe-0/2/0.0 {
            link-protection;
        }
        interface xe-0/1/0.0 {
            link-protection;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path vpls-GOLD-p2mp-template {
            template;
            optimize-timer 50;
            link-protection;
            p2mp;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE5 {
            to 192.0.2.5;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group to-RR {
            type internal;
            local-address 192.0.2.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/0/0.1;
        route-distinguisher 192.0.2.1:1;
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    vpls-GOLD-p2mp-template;
                }
            }
        }
        vrf-target target:65000:1;
        protocols {
            vpls {
                site-range 8;
                no-tunnel-services;
                site CE1 {
                    site-identifier 1;
                    multi-homing;
                    site-preference primary;
                    interface ge-1/0/0.1;
                }
            }
        }
    }
}
```

The relevant sample configuration for Router PE2 follows.

PE2 Router

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ethernet;
}
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.2/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.4.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/3 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    fxp0 {
        apply-groups [ re0 re1 ];
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.2/32;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path vpls-GOLD-p2mp-template {
            template;
            optimize-timer 50;
            link-protection;
            p2mp;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE5 {
            to 192.0.2.5;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group to-RR {
            type internal;
            local-address 192.0.2.2;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/0/3.1;
        route-distinguisher 192.0.2.2:10;
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    vpls-GOLD-p2mp-template;
                }
            }
        }
        vrf-target target:65000:1;
        protocols {
            vpls {
                site-range 8;
                no-tunnel-services;
                site CE1 {
                    site-identifier 1;
                    multi-homing;
                    site-preference backup;
                    interface ge-1/0/3.1;
                }
            }
        }
    }
```

RELATED DOCUMENTATION

Example: NG-VPLS Using Point-to-Multipoint LSPs | **1034**

Next-Generation VPLS Point-to-Multipoint Forwarding Overview | **1028**

Next-Generation VPLS for Multicast with Multihoming Overview | **994**

CHAPTER 37

Configuring Point-to-Multipoint LSPs

IN THIS CHAPTER

- Next-Generation VPLS Point-to-Multipoint Forwarding Overview | **1028**
- Example: NG-VPLS Using Point-to-Multipoint LSPs | **1034**
- Flooding Unknown Traffic Using Point-to-Multipoint LSPs in VPLS | **1076**
- Example: Configuring Ingress Replication for IP Multicast Using MBGP MVPNs | **1081**
- Mapping VPLS Traffic to Specific LSPs | **1099**

Next-Generation VPLS Point-to-Multipoint Forwarding Overview

IN THIS SECTION

- Next-Generation VPLS Point-to-Multipoint Forwarding Applications | **1029**
- Implementation | **1033**

VPLS is a Layer 2 solution for efficiently sending multicast traffic over a multiprotocol label switching (MPLS) core.

VPLS emulates the broadcast domain of a LAN across an MPLS network cloud. Traditional MPLS implementations of VPLS require that all participating ingress provider edge (PE) routers make separate copies of each broadcast or multicast packet to send to all other PE routers that are part of the VPLS site for the same extended LAN. In a large virtual private network (VPN), replication overhead can be significant for each ingress router and its attached core-facing links.

Juniper Networks has several important VPLS enhancements that provide a solution for the replication overhead issue:

- Point-to-multipoint LSP support provides efficient distribution of multicast traffic such as IP-based television (IPTV).

- Multihoming support integrates the path selection capability of BGP with VPLS to allow a customer edge (CE) Ethernet switch to have a backup path across the network.

This document explains the use of point-to-multipoint LSPs in the MPLS core as an alternative to ingress replication. Point-to-multipoint LSPs enable ingress routers to send only one copy of each packet into the MPLS cloud. Each PE router maintains a point-to-multipoint tree so traffic can be efficiently sent to all VPN sites. This process requires the fewest possible replications of the packets and does the replication at the most optimal points in the network.

The benefits of this approach are:

- Conservation of bandwidth
- Increased PE router efficiency
- Improved traffic engineering for flows of flooded traffic
- Manual control or several levels of automatic operation
- Simplified multicast optimization, which is ideal for IPTV or network access wholesale

The Internet Engineering Task Force (IETF) supports two standardized VPLS implementations: *RFC 4761: Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling* and *RFC 4762: Virtual Private LAN Service (VPLS) Using LDP Signaling*.

Juniper Networks has implemented VPLS solutions based on both RFCs. BGP-based VPLS is the superior solution, but LDP-based VPLS is supported for those service providers that have already deployed this alternative.

For a detailed technology overview of LDP-BGP VPLS interworking see *LDP-BGP VPLS Interworking at* https://www.juniper.net/us/en/local/pdf/whitepapers/2000282-en.pdf.

Next-Generation VPLS Point-to-Multipoint Forwarding Applications

VPLS provides a multipoint-to-multipoint Ethernet service that can span one or more metro areas and provides connectivity between multiple sites as if these sites were attached to the same Ethernet LAN.

VPLS uses an IP and MPLS service provider infrastructure. From a service provider’s point of view, use of IP and MPLS routing protocols and procedures instead of the Spanning Tree Protocol (STP), and MPLS labels instead of VLAN IDs, significantly improves the scalability of the VPLS service.

VPLS Protocol Operation

VPLS carries Ethernet traffic across a service provider network, so it must mimic an Ethernet network in some ways. When a PE router configured with a VPLS routing instance receives a packet from a CE device, it first determines whether it knows the destination of the VPLS packet. If it does, it forwards the packet to the appropriate PE router or CE device. If it does not, it broadcasts the packet to all the other

PE routers and CE devices that are members of that VPLS routing instance. In both cases, the CE device receiving the packet must be different from the one sending the packet.

When a PE router receives a packet from another PE router, it first determines whether it knows the destination of the VPLS packet. If the destination is known, the PE router either forwards the packet or drops it, depending on whether the destination is a local or remote CE device. The PE router has three options (scenarios):

- If the destination is a local CE device, the PE router forwards the packet to it.
- If the destination is a remote CE device (connected to another PE router), it discards the packet.
- If it cannot determine the destination of the VPLS packet, the PE router floods it to its attached CE devices.

A VPLS can be directly connected to an Ethernet switch. Layer 2 information gathered by an Ethernet switch, such as media access control (MAC) addresses and interface ports, is included in the VPLS routing instance table. However, instead of all VPLS interfaces being physical switch ports, the router allows remote traffic for a VPLS instance to be delivered across an MPLS LSP and arrive on a virtual port. The virtual port emulates a local, physical port. Traffic can be learned, forwarded, or flooded to the virtual port in almost the same way as traffic sent to a local port.

The VPLS routing table is populated with MAC addresses and interface information for both physical and virtual ports. One difference between a physical port and a virtual port is that on a virtual port, the router captures the outgoing MPLS label used to reach the remote site and an incoming MPLS label for VPLS traffic received from the remote site. The virtual port is generated dynamically on a Tunnel Services PIC when you configure VPLS on a router that is using a Tunnel Services PIC. A Tunnel Services PIC is required to configure a VPLS router on some routers running Junos OS.

If your router has an Enhanced FPC installed, you can configure VPLS without a Tunnel Services PIC. To do so, you use a label-switched interface (LSI) to provide VPLS functionality. An LSI MPLS label is used as the inner label for VPLS. This label maps to a VPLS routing instance. On the PE router, the LSI label is stripped and then mapped to a logical LSI interface. The Layer 2 Ethernet frame is then forwarded using the LSI interface to the correct VPLS routing instance. To configure VPLS on a router without a Tunnel Services PIC, include the `no-tunnel-services` statement.

One restriction on flooding behavior in VPLS is that traffic received from remote PE routers is never forwarded to other PE routers. This restriction helps prevent loops in the core network. This also means that the core network of PE routers must be fully meshed. Additionally, if a CE Ethernet switch has two or more connections to the same PE router, you must enable the Spanning Tree Protocol (STP) on the CE switch to prevent loops.

Point-to-Multipoint Implementation

In next-generation VPLS, point-to-multipoint LSPs are used to flood broadcast, multicast, and unknown unicast traffic across a VPLS core network to all the PE routers. This is more efficient in terms of bandwidth utilization between the PE router and provider (P) router.

If point-to-multipoint LSPs are not being used, the PE router needs to forward multiple copies of broadcast, multicast, and unknown unicast packets to all PE routers. If point-to-multipoint LSPs are used, the PE router floods one copy of each packet to the P router, where it is replicated close to the egress router.

NOTE: For next-generation VPLS, both point-to-point LSPs and point-to-multipoint LSPs are needed between the PE routers.

In VPLS, point-to-multipoint LSPs are only used to transport broadcast frames, multicast frames, and unicast frames with an unknown destination MAC address. All other frames are still transported using point to-point LSPs. This structure is much more efficient for bandwidth use, particularly near the source of the broadcast, multicast, and unknown frames. However, it also results in more state in the network because each PE router is the ingress of one point-to-multipoint LSP that touches all other PE routers and one point-to-point LSP going to each of the other PE routers.

Enabling point-to-multipoint LSPs for any VPLS instance starts the flooding of unknown-unicast, broadcast, and multicast traffic using point-to-multipoint LSPs.

For each VPLS instance, a PE router creates a dedicated point-to-multipoint LSP. Whenever VPLS discovers a new neighbor through BGP, a source-to-leaf sub-LSP is added for this neighbor in the point-to-multipoint LSP instance.

If there are *n* PE routers in the VPLS instance, then the discovery of a new neighbor through BGP creates *n* point-to-multipoint LSPs in the network, where each PE router is the root of the tree and the rest of the *n-1* PE routers are leaf nodes (or source-to-leaf sub-LSPs).

Each point-to-multipoint LSP created by PE routers can be identified using an RSVP-traffic engineering point-to-multipoint session object, which is passed as a provider multicast service interface (PMSI) tunnel attribute by BGP while advertising VPLS routes. Using this tunnel attribute, incoming source-to-leaf sub-LSP add request messages (RSVP-path message) can be associated with the right VPLS instance and originator PE router. As a result, label allocation is done in such a way that when traffic arrives on the LSP, it is not only terminated on the right VPLS instance, but the originator PE router is also identified so that source MAC addresses can be learned.

Point-to-multipoint LSPs can be enabled incrementally on any PE router that is part of a specific VPLS instance. This means a PE router that has this feature uses point-to-multipoint LSPs to flood traffic, whereas other PE routers in the same VPLS instance can use ingress replication to flood the traffic. However, when point-to-multipoint LSPs are enabled on any PE router, make sure that all the PE routers that are part of the same VPLS instance also support this feature.

NOTE: Penultimate-hop popping (PHP) is disabled for point-to-multipoint LSPs terminating in a VPLS instance.

Limitations of Point-to-Multipoint LSPs

When implementing point-to-multipoint LSPs remember the following limitations:

- There is no mechanism to allow only multicast traffic to go over the point-to-multipoint LSP.
- Point-to-multipoint LSPs do not support inter-AS traffic. Only intra-AS traffic is supported.
- Point-to-multipoint LSPs do not support graceful restart for ingress LSPs. This also affects VPLS when flooding is done using point-to-multipoint LSPs.
- The same point-to-multipoint LSP cannot be shared across multiple VPLS instances.
- When this feature is enabled, ingress PE routers use only point-to-multipoint LSPs for flooding. The router initiates the creation of source-to-leaf sub-LSPs for each PE router that is part of the same VPLS instance. Any PE router for which this source-to-leaf sub-LSP fails to come up does not receive any flooded traffic from the ingress PE router.
- It is possible that flooding of unknown unicast traffic over point-to-multipoint LSPs may lead to packet reordering, because as soon as learning is done, unicast traffic is sent out using point-to-point pseudowire LSPs.
- Static LSPs and LSPs configured using the `label-switched-path-template` statement cannot be configured at the same time.
- When an LSP is configured using the `static-lsp` statement, a point-to-multipoint LSP is created statically to include all neighbors in the VPLS instance.

Before enabling the point-to-multipoint LSP feature on any PE router, make sure that all the other PE routers that are part of the same VPLS instance are upgraded to a Junos OS Release that supports it. If a router in the VPLS instance does not support point-to-multipoint LSPs, it may lose all the traffic sent on the point-to-multipoint LSP. Therefore, do not enable this feature if there is a single router in a VPLS instance that is not capable of supporting this feature, either because it is not running the appropriate Junos OS Release or because it is a router from a vendor that does not support this feature.

Simultaneous Transit and Egress Router Operation

A PE router that plays the role of both an MPLS transit router and an MPLS egress router can do so by receiving either one or two copies of a packet to fulfill each of its roles.

To fulfill both roles while using only a single copy of a packet, some Juniper Networks routers require a Tunnel Services PIC configured with virtual tunnel (vt) interfaces and ultimate-hop popping must be enabled. With a virtual tunnel interface and ultimate-hop popping, a single copy of the received packet is forwarded beyond the PE router to fulfill the transit router role and is also consumed internally by the virtual tunnel interface to fulfill the egress router role.

If a label-switched interface (LSI) *logical interface* is used, then two copies of each packet must be received on the point-to-multipoint LSP, one to fulfill the transit router role and one to fulfill the egress router role.

Implementation

Some implementations of VPLS use ingress replication. Ingress replication is simple but inefficient. It sends multiple copies of the same packet on a link, especially the PE-P link. This causes wasted bandwidth when there is a heavy broadcast and multicast traffic.

As shown in the sample network in Figure 76 on page 1033 the ingress PE router makes three copies of every broadcast, multicast, and flooded packet for each VPLS instance.

Figure 76: Ingress Replication

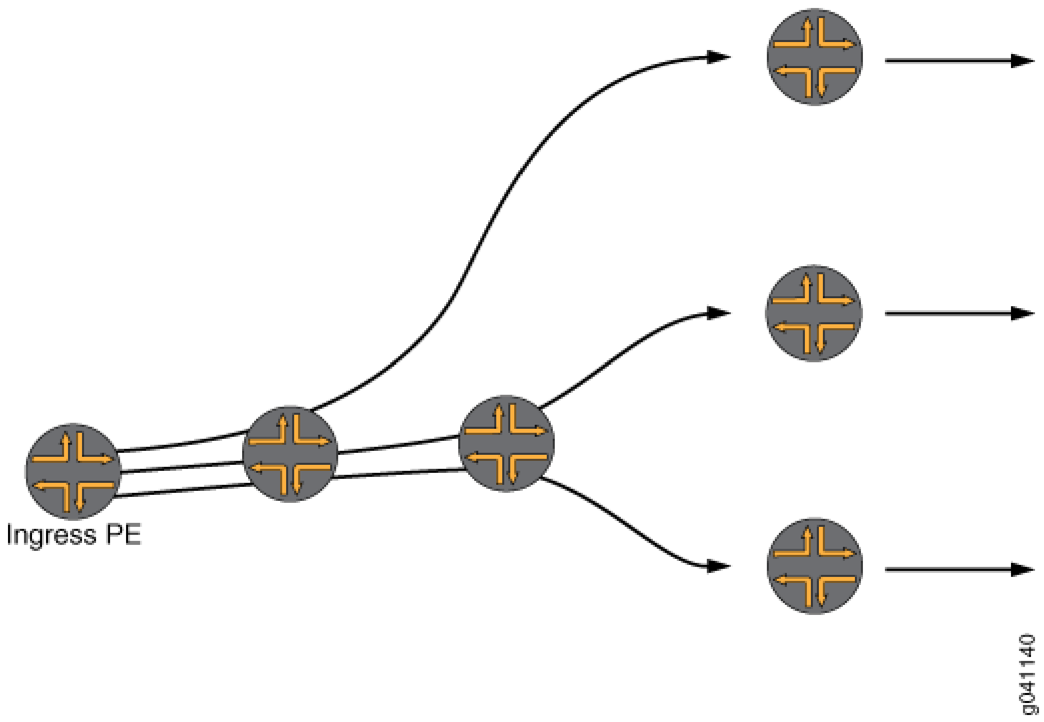
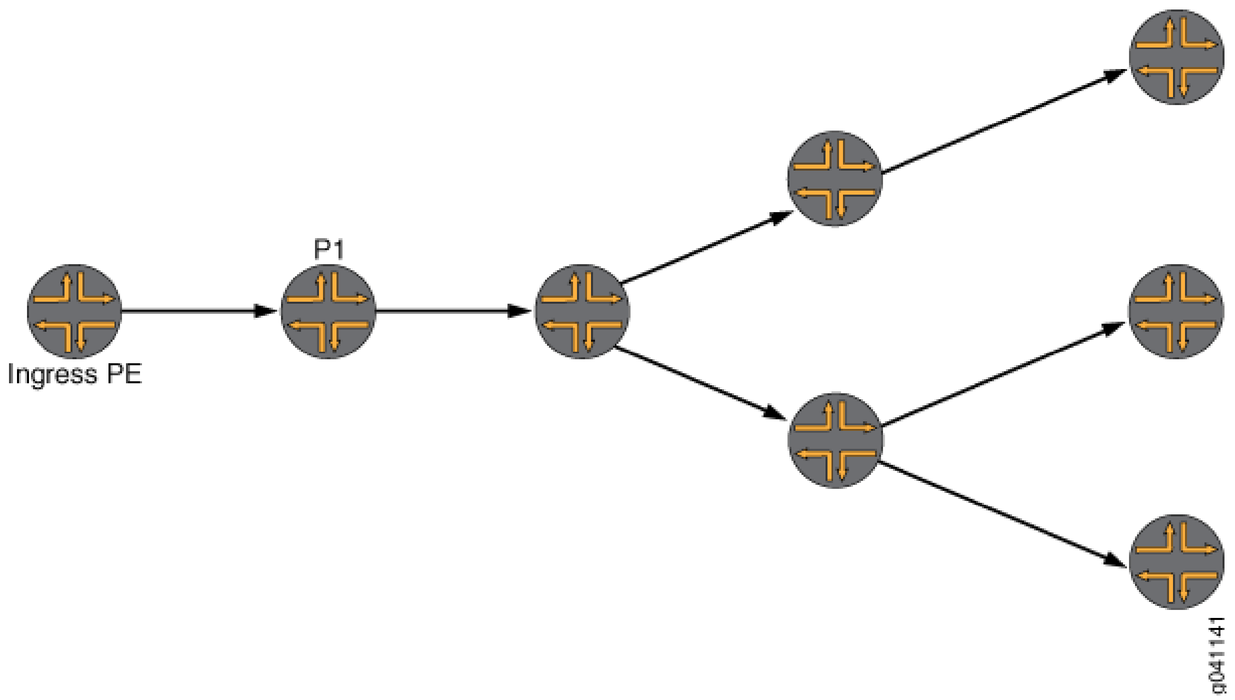


Figure 77 on page 1034 shows how a point-to-multipoint LSP works for multicast.

In a VPLS using point-to-multipoint LSPs, the ingress PE router sends a single copy of the multicast packet to Router P1. Router P1 makes two copies for this point-to-multipoint LSP. Each of the other P routers also makes multiple copies of the packet. This moves replication closer to the endpoints and results in significant improvements in the network bandwidth utilization.

Figure 77: Point-to-Multipoint Replication



RELATED DOCUMENTATION

Example: NG-VPLS Using Point-to-Multipoint LSPs | **1034**

Example: NG-VPLS Using Point-to-Multipoint LSPs

IN THIS SECTION

- Requirements | **1035**
- Overview and Topology | **1035**
- Configuration | **1038**

This example shows how to configure next-generation VPLS (NG_VPLS) using point-to-multipoint LSPs. The topology is shown in Figure 78 on page 1036 and Figure 79 on page 1037. This example is organized in the following sections:

Requirements

Table 24 on page 1035 lists the hardware that is used and the software that is required for this example:

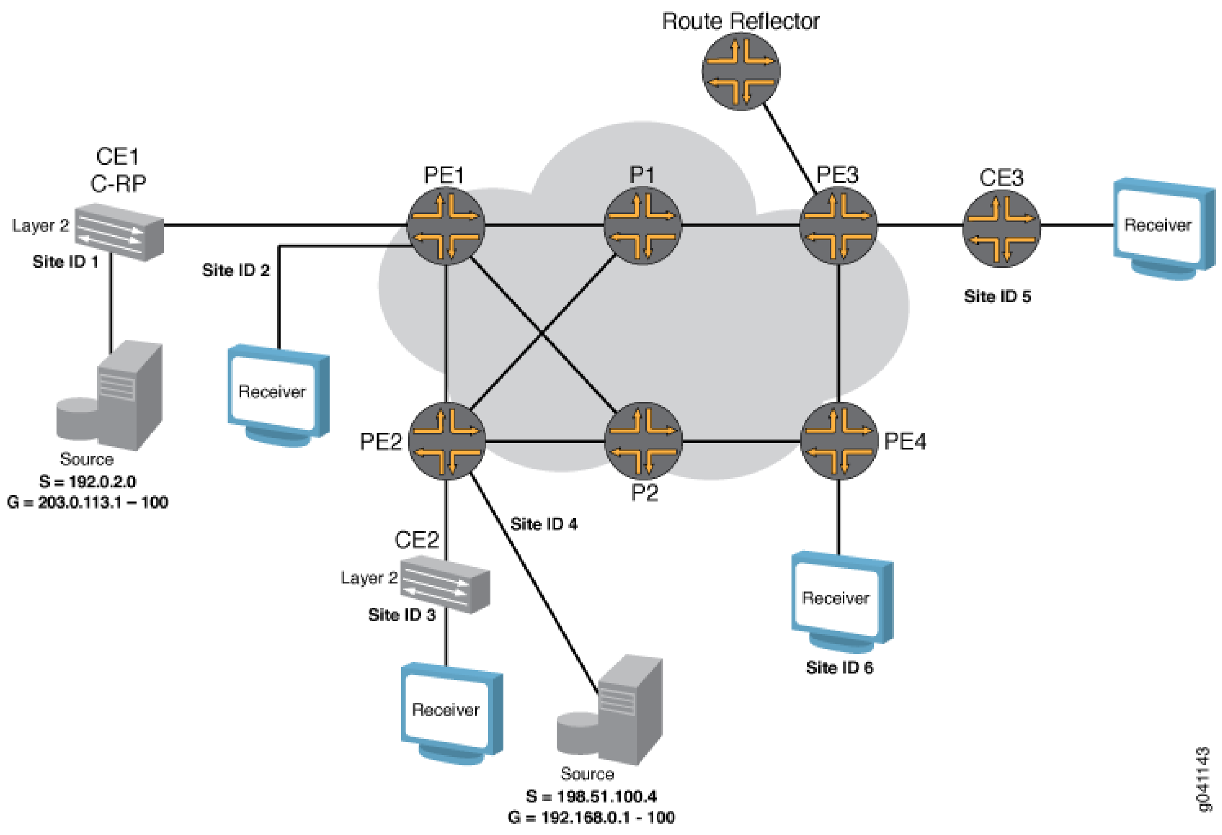
Table 24: Hardware and Software Used

Equipment	Components	Software
Six MX Series 5G Universal Routing Platforms	DPC-4 10GE-X, DPC-40 1GE-X	Junos OS Release 9.3R4 or later
One T Series Core Router	FPC3, 10GE-Xenpak	Junos OS Release 9.3R4 or later
Eight EX4200 Ethernet Switches	EX4200 virtual switches	Junos OS Release 9.3R4 or later
One M7i Multiservice Edge Router	Gigabit Ethernet interfaces	Junos OS Release 9.3R4 or later

Overview and Topology

The logical topology of the NG-VPLS example is shown in Figure 78 on page 1036.

Figure 78: Logical Topology of NG-VPLS Using Point-to-Multipoint LSPs

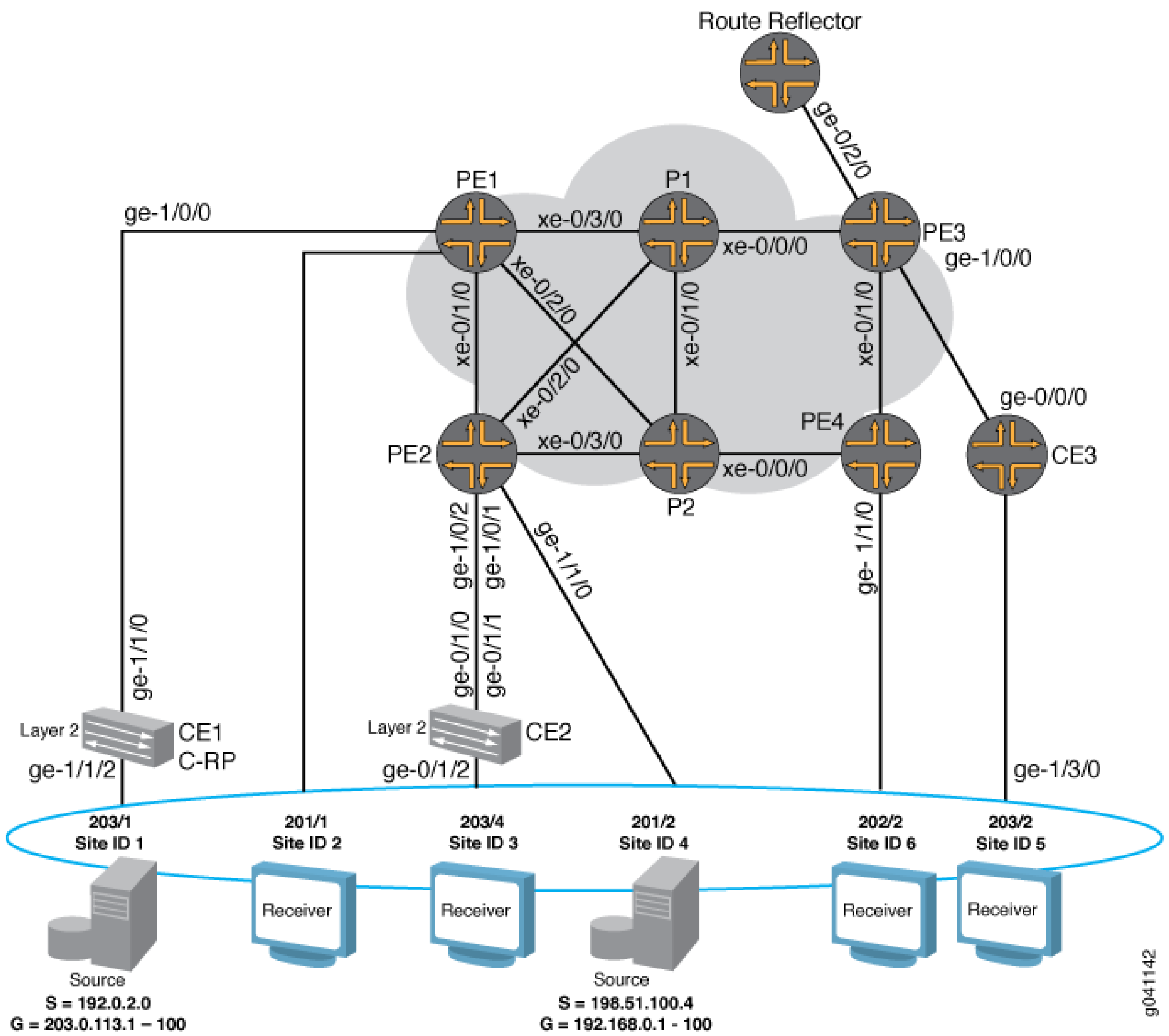


The routers in this example are preconfigured with the following:

- OSPF area 0 is configured on all the PE routers and P routers with traffic engineering enabled.
- All of the core-facing interfaces are configured with the `mpls` protocol address family.
- The RSVP and MPLS protocols are enabled for all the core-facing interfaces.
- All the MX Series routers have their network services mode set to Ethernet. The network services mode is configured by including the `network-services` statement and specifying the `ethernet` option.
- All the PE routers are configured for autonomous system `65000`.

The physical topology of the NG-VPLS example is shown in Figure 79 on page 1037. The topology consists of six MX Series routers connected with redundant links in the core. Four MX Series routers are acting as PE routers and two are core routers.

Figure 79: Physical Topology of NG-VPLS Using Point-to-Multipoint LSPs



Note the following topology details:

- A route reflector is configured in the topology to reflect the family `l2-vpn` routes to all the PE routers for BPG-VPLS.
- The GOLD VPLS routing instance is configured with two sites in each of the PE routers.
- One GOLD site is connected to the CE router and the other one is directly connected to the test equipment on each PE router.
- The `no-tunnel-services` statement is included in the GOLD VPLS instance to enable the use of LSI interfaces for VPLS tunnel services.
- Router CE1 and Router CE2 are EX Series Virtual Chassis switches acting as CE routers.

- Router CE3 is an M7i router acting as a CE router.
- Two multicast sources are configured. One is connected to Router CE1 (Site 1) and the other to Router PE2 (Site 4) to simulate different scenarios.
- Router CE1 is configured as the rendezvous point (RP).
- Unicast traffic is enabled on all the test equipment ports and is sent to all the sites in the GOLD VPLS instance.

Configuration

IN THIS SECTION

- Configuring the PE Router Interfaces | 1038
- Configuring a Route Reflector for all PE Routers for BGP-Based VPLS | 1040
- Establishing BGP-Based VPLS with a Route Reflector | 1042
- Configuring Point-to-Point LSPs Between PE Routers | 1043
- Configuring Dynamic and Static Point-to-Multipoint LSPs Between PE Routers | 1044
- Configuring Point-to-Multipoint Link Protection | 1045
- Configuring a BGP-Based VPLS Routing Instance for NG-VPLS | 1047
- Configuring Tunnel Services for VPLS | 1050
- Verifying the Control Plane | 1051
- Verifying the Data Plane | 1061
- Results | 1068

This example shows how to configure next-generation VPLS using point-to-multipoint LSPs. It is organized in the following sections:

Configuring the PE Router Interfaces

Step-by-Step Procedure

On the customer-facing PE interfaces, enable VLAN tagging, configure the encapsulation type, and enable the VPLS address family. There are four possible interface encapsulations for VPLS routing instances that you can choose depending on your needs.

1. If your network requires that each logical interface on the PE router-to-CE router link be configured to only accept packets with VLAN ID `1000`, include the `vlan-tagging` statement, include the `encapsulation` statement, and specify `vlan-vpls` as the encapsulation type. Also include the `vlan-id` statement and specify `1000` as the VLAN ID.

   ```
   [edit interfaces]
   ge-1/1/0 {
       vlan-tagging;
       encapsulation vlan-vpls;
       unit 1 {
           encapsulation vlan-vpls;
           vlan-id 1000;
           family vpls;
       }
   }
   ```

   With this configuration, you can configure multiple logical interfaces with different VLAN IDs and associate each logical interface with a different routing instance.

2. If your network requires each physical interface on the PE router to CE router link to be configured to use the entire Ethernet port as part of a single VPLS instance, include the `encapsulation` statement, and specify `ethernet-vpls` as the encapsulation type.

   ```
   [edit interfaces]
   ge-1/2/0 {
       encapsulation ethernet-vpls;
       unit 0 {
           family vpls;
       }
   }
   ```

   With this encapsulation mode, you cannot create multiple logical units (VLANs).

3. If your network requires that each logical interface of the single physical interface on the PE router to CE router link be configured to use a mix of different encapsulations, include the `encapsulation` statement, and specify `flexible-ethernet-services` as the encapsulation type at the `[edit interfaces` *`interface-name`*`]` hierarchy level. Also include the `encapsulation` statement, and specify `vlan-vpls` or `vlan-ccc` as the encapsulation type at the `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`*`]` hierarchy level.

   ```
   [edit interfaces]
   ge-1/2/0 {
   ```

```
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 1 {
        encapsulation vlan-vpls;
    }
    unit 2 {
        encapsulation vlan-ccc;
    }
}
```

4. If your network requires support for using a mix of single and dual tagged VLANs configured in different logical interfaces on a single physical interface, include the `encapsulation` statement, and specify `flexible-vlan-tagging` as the encapsulation type.

5. Configure the core-facing CE router interfaces. The CE router and PE router logical interface configuration must match encapsulation types and VLAN IDs. Typically the IP address is configured on the core-facing CE router interfaces if the CE device is a router and terminates the Layer 2 domain into the Layer 3 network. In this example, the interface is configured for single tagging with a VLAN ID of `1000`.

```
[edit interfaces]
ge-1/1/0 {
    vlan-tagging;
    unit 1 {
        vlan-id 1000;
        family inet {
            address 198.51.100.4/24;
        }
    }
}
```

Configuring a Route Reflector for all PE Routers for BGP-Based VPLS

Step-by-Step Procedure

Configuring a route reflector is the preferred method to enable any BGP-based service offerings. Configuring a route reflector avoids the requirement for a full mesh of BGP peer sessions, and it scales well. BGP redundancy can be achieved using multiple route reflectors in a single cluster.

1. To enable BGP to carry Layer 2 VPN and VPLS NLRI messages, create a peer group, include the `family` statement, specify the `l2vpn` option, and include the `signaling` statement. To configure the route

reflector cluster and complete the BGP peer sessions, include the `cluster` statement and specify the IP address for the cluster ID. Then include the `neighbor` statement and specify the IP address of the PE routers that are BGP client peers in the cluster.

```
[edit protocols]
bgp {
    group RR {
        type internal;
        local-address 192.0.2.7;
        family l2vpn {
            signaling;
        }
        cluster 192.0.2.7;
        neighbor 192.0.2.1;  # To PE1
        neighbor 192.0.2.2;  # To PE2
        neighbor 192.0.2.3;  # To PE3
        neighbor 192.0.2.4;  # To PE4
    }
}
```

2. Configure OSPF and enable traffic engineering on the route reflector to create the Constrained Shortest Path First (CSPF) database for the egress LSPs terminating from the PE routers.

```
[edit protocols]
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

3. Enable the MPLS and RSVP protocols on all interfaces connected to the MPLS core. This terminates the RSVP egress LSPs from the PE routers.

```
[edit protocols]
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

Establishing BGP-Based VPLS with a Route Reflector

Step-by-Step Procedure

For BGP-based VPLS, all PE routers need to have a full mesh of BGP peer sessions with each other or have a single peer with the route reflector. The route reflector reflects the routes received from the other PE routers. In this example, the PE router is configured to establish a peer relationship with the route reflector.

1. To have all the PE routers establish a BGP client peer session with the route reflector, create an internal peer group, include the `local-address` statement, and specify the IP address of the PE router. Also include the `neighbor` statement, and specify the IP address of the route reflector. To enable BGP to carry Layer 2 VPN and VPLS NLRI messages, include the `family` statement, specify the `l2vpn` option, and include the `signaling` statement.

   ```
   [edit protocols]
   bgp {
       group to-RR {
           type internal;
           local-address 192.0.2.1;
           family l2vpn {
               signaling;
           }
           neighbor 192.0.2.7;  # To the route reflector
       }
   }
   ```

2. Configure a point-to-point RSVP LSP from the PE routers to the route reflector. To create the LSP, include the `label-switched-path` statement, give the LSP a meaningful name, include the `to` statement

and specify the IP address of the route reflector as the LSP end point. This LSP is needed to resolve the BGP next hops in the `inet.3` routing table for the routes received from the route-reflector.

```
[edit protocols]
mpls {
    label-switched-path to-RR {
        to 192.0.2.7;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

Configuring Point-to-Point LSPs Between PE Routers

Step-by-Step Procedure

In next-generation VPLS, point-to-multipoint LSPs are only used to transport broadcast, multicast, and unknown unicast frames. All other frames are still transported using point-to-point RSVP LSPs. This is a more efficient use of bandwidth, particularly near the source of the unknown, broadcast, and multicast frames. The trade-off is more state in the network, because each PE router is the ingress of one point-to-multipoint LSP that touches all other PE routers, and *n* point-to-point LSPs are needed, one going to each of the other PE routers.

1. To create a point-to-point LSP, include the `label-switched-path` statement, give the LSP a meaningful name, include the `to` statement, and specify the IP address of the other PE router as the LSP endpoint. The example shows the configuration of LSPs from Router PE1 to Routers PE2, PE3, and PE4.

```
[edit protocols]
mpls {
    label-switched-path to-PE2 {
        to 192.0.2.2;
    }
    label-switched-path to-PE3 {
        to 192.0.2.3;
    }
    label-switched-path to-PE4 {
        to 192.0.2.4;
    }
}
```

Configuring Dynamic and Static Point-to-Multipoint LSPs Between PE Routers

Step-by-Step Procedure

This procedure describes how to enable the creation of dynamic point-to-multipoint LSPs and how to configure static point-to-multipoint LSPs. On a router configured with static point-to-multipoint LSPs, the LSPs come up immediately. On a router configured with dynamic point-to-multipoint LSPs, the LSP comes up only after receiving BGP neighbor information from the route reflector or from the other PE routers participating in the VPLS domain.

For each VPLS instance, a PE router with dynamic point-to-multipoint LSPs enabled creates a dedicated point-to-multipoint LSP based on the point-to-multipoint template. Whenever VPLS discovers a new neighbor through BGP, a sub-LSP for this neighbor is added to the point-to-multipoint LSP.

If there are *n* PE routers in the VPLS instance then the router creates *n* point-to-multipoint LSPs in the network where each PE router is the root of the tree and includes the rest of the *n-1* PE routers as leaf nodes connected through a source-to-leaf sub-LSP.

1. In this step, you configure Router PE1 and Router PE2 to use a dynamic point-to-multipoint LSP template for LSP creation. When these routers receive a new BGP route advertised from the route reflector for a new neighbor, they create a point-to-multipoint sub-LSP to that neighbor. To create the dynamic point-to-multipoint LSP template, include the `label-switched-path` statement, give the LSP template a meaningful name, include the `template` statement and include the `p2mp` statement. Also enable link protection and configure the optimize timer to periodically reoptimize the LSP path.

```
[edit protocols]
mpls {
    label-switched-path vpls-GOLD-p2mp-template {
        template;  # identify as a template
        optimize-timer 50;
        link-protection;  # link protection is enabled on point-to-multipoint LSPs
        p2mp;
    }
}
```

2. In this step, you configure static point-to-multipoint LSPs. Creating static point-to-multipoint LSPs is similar to creating point-to-point LSPs, except you can also configure other RSVP parameters under each point-to-multipoint LSP.

To create static point-to-multipoint LSPs, include the `label-switched-path` statement, give the LSP a meaningful name, include the `to` statement, and specify the IP address of the PE router that is the endpoint of the LSP. Also include the `p2mp` statement and specify a pathname.

```
[edit protocols]
mpls {
    label-switched-path to-pe2 {
        to 192.0.2.2;
        p2mp vpls-GOLD;
    }
    label-switched-path to-pe3 {
        to 192.0.2.3;
        p2mp vpls-GOLD;
    }
    label-switched-path to-pe1 {
        to 192.0.2.1;
        p2mp vpls-GOLD;
    }
}
```

Configuring Point-to-Multipoint Link Protection

Step-by-Step Procedure

Point-to-multipoint LSPs only support RSVP link protection for traffic engineering. Node protection is not supported. Link protection is optional, but it is the recommended configuration for most networks.

1. To enable link protection on the core-facing interfaces, include the `link-protection` statement at the `[edit protocols rsvp interface` *`interface-name`*`]` hierarchy level.

```
[edit protocols]
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
    interface xe-0/3/0.0 {
        link-protection;
    }
    interface xe-0/2/0.0 {
        link-protection;
    }
    interface xe-0/1/0.0 {
        link-protection;
    }
}
```

2. Enable the point-to-multipoint LSP to use the RSVP link protection feature. Link-protection can be configured for both static point-to-multipoint and dynamic point-to-multipoint LSPs that use a template.

   For static point-to-multipoint LSPs, configure each branch sub-LSP. To enable link protection, include the `link-protection` statement at the `[edit protocols mpls label-switched-path` *`label-switched-path-name`*`]` hierarchy level.

```
[edit protocols mpls label-switched-path]
label-switched-path to-pe2 {
    to 192.0.2.2;
    link-protection;
    p2mp vpls-GOLD;
}
label-switched-path to-pe3 {
    to 192.0.2.3;
    link-protection;
    p2mp vpls-GOLD;
}
label-switched-path to-pe1 {
    to 192.0.2.1;
    link-protection;
    p2mp vpls-GOLD;
}
```

3. For dynamic point-to-multipoint LSPs using a template, only the template needs to have link protection configured. All the point-to-multipoint branch LSPs that use the template inherit this configuration.

   To enable link protection for dynamic point-to-multipoint LSPs, include the `link-protection` statement at the `[edit protocols mpls label-switched-path` *`label-switched-path-name`*`]` hierarchy level.

```
[edit protocols mpls label-switched-path]
label-switched-path vpls-GOLD-p2mp-template {
    template;
    optimize-timer 50;
    link-protection;
    p2mp;
}
```

Configuring a BGP-Based VPLS Routing Instance for NG-VPLS

Step-by-Step Procedure

For NG-VPLS, the routing-instance configuration is similar to that for a regular VPLS routing instance. The routing instance defines the VPLS site and creates the VPLS connection. The following parameters are configured.

- Instance Type – VPLS.
- Interface – The interface connecting to the CE router.
- Route Distinguisher – Each routing instance you configure on a PE router must have a unique route distinguisher. The route distinguisher is used by BGP to distinguish between potentially identical network reachability information (NLRI) messages received from different VPNs. We recommend that you use a unique route distinguisher for each routing instance on each PE so that you can determine which PE originated the route.
- VRF Target – Configuring a VRF target community using the `vrf-target` statement causes default VRF import and export policies to be generated that accept imported routes and tag exported routes with the specified target community.
- Protocols – Configure the VPLS protocol as described in the following procedure.

1. To configure the NG-VPLS routing instance, include the `routing-instances` statement and specify the instance name. Also include the `instance-type` statement and specify `vpls` as the type. Include the `route-distinguisher` statement and specify a route distinguisher that is unique throughout all VPNs configured on the router. Configure a VRF route target by including the `vrf-target` statement and specify the route target. The route target exported by one router must match the route target imported by another router for the same VPLS.

```
[edit]
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/0/0.1;
        interface ge-1/1/0.1;
        route-distinguisher 192.0.2.1:1;
        vrf-target target:65000:1;
    }
}
```

2. To use a point-to-multipoint LSP for VPLS flooding, configure an LSP under the VPLS routing instance.

   To configure the point-to-multipoint LSP for VPLS flooding, include the `label-switched-path-template` statement and specify the name of the LSP template at the `[edit routing-instances` *`routing-instances-name`* `provider-tunnel rsvp-te]` hierarchy level.

   ```
   [edit]
   routing-instances {
       GOLD {
           instance-type vpls;
           interface ge-1/0/0.1;
           interface ge-1/1/0.1;
           route-distinguisher 192.0.2.1:1;
           provider-tunnel {
               rsvp-te {
                   label-switched-path-template {
                       vpls-GOLD-p2mp-template;
                   }
               }
           }
           vrf-target target:65000:1;
       }
   }
   ```

3. Configuring the VPLS protocol enables the VPLS between different sites in the VPLS domain. Multiple sites can be configured under a single VPLS routing instance, but note that the lowest site ID is used to build the VPLS pseudowire to the other PE routers, and the label block associated with the lowest site ID is advertised. The following parameters are configured for the VPLS protocol:

   - Site – Name of the VPLS site.
   - Site Range – Maximum site ID allowed in the VPLS. The site range specifies the highest-value site ID allowed within the VPLS, not the number of sites in the VPLS.
   - Site Identifier – Any number between 1 and 65,534 that uniquely identifies the VPLS site. This is also referred as the VE-ID in the relevant RFC.
   - PE-CE Interface – The interface participating in this site.

- Tunnel services for VPLS – If you do not configure any tunnel interface at the `[edit protocol vpls tunnel-services]` hierarchy, the router uses any tunnel interface available on the router for VPLS.
- No-tunnel-services – If you include the `no-tunnel-services` statement, the router uses a label-switched interface (LSI) for the tunnel services for that VPLS instance.
- Mac Table Size – The size of the VPLS media access control (MAC) address table. The default is 512 addresses and the maximum is 65,536. When the table is full, new MAC addresses are no longer added to the table.

To configure the VPLS protocol, include the `vpls` statement at the `[edit routing-instances` *routing-instance-name* `protocols]` hierarchy level. To configure the site range, include the `site-range` statement and specify the highest-value site ID allowed within the VPLS. To cause the router to use an LSI interface, include the `no-tunnel-services` statement. To create a VPLS site, include the `site` statement and specify a site name. Also include the `site-identifier` statement and specify the site ID. Then include the `interface` statement and specify the interface name for the interface connected to the CE device.

```
[edit]
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/0/0.1;
        interface ge-1/1/0.1;
        route-distinguisher 192.0.2.1:1;
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    vpls-GOLD-p2mp-template;
                }
            }
        }
        vrf-target target:65000:1;
        protocols {
            vpls {
                site-range 8;
                no-tunnel-services;
                site CE1 {
                    site-identifier 1;
                    interface ge-1/0/0.1;
                }
                site Direct {
                    site-identifier 2;
                    interface ge-1/1/0.1;
                }
            }
        }
    }
}
```

Configuring Tunnel Services for VPLS

Step-by-Step Procedure

A tunnel interface is needed for VPLS configuration to encapsulate the originating traffic, and to de-encapsulate the traffic coming from a remote site. If the tunnel interface is not configured, the router selects one of the available tunnel interfaces on the router by default. There are three methods available in Junos OS to configure this tunnel interface.

- To specify a virtual tunnel interface to be used as the primary device for tunneling, include the `primary` statement, and specify the virtual tunnel interface to be used at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls tunnel-services]` hierarchy level.

```
[edit routing-instances routing-instance-name]
protocols {
    vpls {
        site-range 8;
        tunnel-services {
            primary vt-1/2/10;
        }
    }
}
```

- To configure the router to use an LSI interface for tunnel services rather than a virtual tunnel interface, include the `no-tunnel-services` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level.

```
[edit routing-instances routing-instance-name]
protocols {
    vpls {
        site-range 8;
        no-tunnel-services;
    }
}
```

- In an MX Series router you must create the tunnel services interface to be used for tunnel services. To create the tunnel service interface, include the `bandwidth` statement and specify the amount of bandwidth to reserve for tunnel services in gigabits per second at the `[edit chassis fpc` *`slot-number`* `pic` *`slot-number`* `tunnel-services]` hierarchy level.

```
[edit chassis]
fpc 1 {
    pic 3 {
        tunnel-services {
            bandwidth 1g;
        }
    }
}
```

Verifying the Control Plane

Step-by-Step Procedure

This section describes `show` command outputs you can use to validate the control plane. It also provides methodologies for troubleshooting. Note the following:

- In this example there are six sites. Router PE1 and Router PE2 have two sites each. Router PE3 and Router PE4 have one site each. All sites are in the GOLD VPLS instance.
- In VPLS if you have multiple sites configured under a single VPLS routing instance, the label block from the site with the lowest site ID is used to establish pseudowires between remote PEs. Note that the data traffic is still sent to those PE router interfaces connected to CE devices that are in one of the following states:
    - LM – Local site ID is not the minimum designated. The local site ID is not the lowest. Therefore the local site ID is not being used to establish pseudowires or distribute VPLS label blocks.
    - RM – Remote site ID is not the minimum designated. The remote site ID is not the lowest. Therefore, the remote site ID is not being used to establish pseudowires or distribute VPLS label blocks.
- For more information about how VPLS label blocks are allocated and used, see *Understanding VPLS Label Blocks Operation*.

1. After the entire configuration is done, you can verify the VPLS connections state.

In the following output, the VPLS connections show the `Up` state for certain sites, and the remaining sites show either the `RM` or `LM` state. This is the expected state in a VPLS implementation on multihoming sites.

In this example, Router PE1 has site `CE1` configured with site ID `1` and site `Direct` configured with site ID `2`. The label block for site `CE1` is advertised to the remote PE routers and used for receiving the data packets from the remote PE routers. In the `show` command output, notice the following:

- Router PE1 uses its lowest site ID, which is site ID `1`. Site ID 1 is used for Device `CE1`.
- Router PE2 uses its lowest site ID, which is site ID `3`. Site ID 3 is used for Device `CE2`.
- Router PE3 and Router PE4 each have a single site configured.

  For site `CE1`, connection site `3` is in the `Up` state and connection site `4` is in the `RM` state.
- For site `Direct`, all the connections are in the `LM` state.
- Site `Direct` has a higher site ID than site `1` on this router.

On Router PE1, use the `show vpls connections` command to verify the VPLS connections state .

```
user@PE1> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection

Legend for interface status
Up -- operational
Dn -- down
```

```
Instance: GOLD
  Local site: CE1 (1)
    connection-site           Type  St     Time last up       # Up trans
    3                         rmt   Up    Oct  6 16:27:23 2009       1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 262171, Outgoing label: 262145
      Local interface: lsi.1049353, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 3
    4                         rmt   RM
    5                         rmt   Up     Oct  6 16:27:27 2009       1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 262173, Outgoing label: 262145
      Local interface: lsi.1049354, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 5
    6                         rmt   Up     Oct  6 16:27:31 2009       1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 262174, Outgoing label: 800000
      Local interface: lsi.1049355, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 6
  Local site: Direct (2)
    connection-site           Type  St     Time last up       # Up trans
    3                         rmt   LM
    4                         rmt   LM
    5                         rmt   LM
    6                         rmt   LM
```

2. On Router PE4, use the `show vpls connections` command to verify the VPLS connections state.

   Verify that site `2` and site `4` are in the `RM` state. This state tells you that the sites are configured with the highest site ID on Router PE1 and Router PE2. Because Router PE4 has only one site configured, it does not have any sites in the `LM` states.

```
user@PE4> show vpls connections
...
Instance: GOLD
  Local site: Direct (6)
    connection-site           Type  St     Time last up      # Up trans
    1                         rmt   Up     Oct  6 16:28:35 2009      1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 262174
      Local interface: vt-1/2/10.1048576, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 6 remote site 1
    2                         rmt   RM
    3                         rmt   Up     Oct  6 16:28:35 2009      1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 800002, Outgoing label: 262150
      Local interface: vt-1/2/10.1048577, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 6 remote site 3
    4                         rmt   RM
    5                         rmt   Up     Oct  6 16:28:35 2009      1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800004, Outgoing label: 262150
      Local interface: vt-1/2/10.1048578, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 6 remote site 5
```

3. On each PE router, use the `show bgp summary` command to verify that the IBGP sessions between the PE routers or between the PE router and the route reflector have been established. The sessions must be operational before the PE routers can exchange any Layer 2 VPN routes. In the example below, also notice that the output from Router PE1 shows that the `bgp.l2vpn.0` and `GOLD.l2vpn.0` routing tables have been created.

```
user@PE1> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table    Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0      4          4          0         0          0          0
Peer               AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State
192.0.2.7      65000         40         39       0       1       15:45 Establ
  bgp.l2vpn.0: 4/4/4/0
  GOLD.l2vpn.0: 4/4/4/0

admin@PE2# run show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table    Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0      4          4          0         0          0          0
inet6.0          0          0          0         0          0          0
inet.0           0          0          0         0          0          0
Peer             AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State
192.0.2.7    65000         43         42       0       0       17:25 Establ
  bgp.l2vpn.0: 4/4/4/0
  GOLD.l2vpn.0: 4/4/4/0
```

4. On Router PE4, use the `show route table` command to verify that there is one Layer 2 VPN route to each of the other PE routers. Router PE3 should have a similar `show` command output.

```
user@PE4> show route table bgp.l2vpn.0
bgp.l2vpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.0.2.1:1:1:1/96
                   *[BGP/170] 00:23:18, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.9.1 via xe-0/0/0.0, label-switched-path to-PE1
192.0.2.1:1:2:1/96
                   *[BGP/170] 00:23:18, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.9.1 via xe-0/0/0.0, label-switched-path to-PE1
192.0.2.2:10:3:1/96
                   *[BGP/170] 00:23:18, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.9.1 via xe-0/0/0.0, label-switched-path to-PE2
192.0.2.2:10:4:1/96
                   *[BGP/170] 00:23:18, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.9.1 via xe-0/0/0.0, label-switched-path to-PE2
192.0.2.3:10:5:1/96
                   *[BGP/170] 00:23:18, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.8.1 via xe-0/1/0.0, label-switched-path to-PE3
```

5. On the route reflector, use the `show bgp summary` command to verify that the router has an IBGP peer session with each of the PE routers.

```
user@RR> show bgp summary
Groups: 2 Peers: 5 Down peers: 1
Table    Tot Paths  Act Paths Suppressed    History Damp State  Pending
bgp.l2vpn.0      6          6          0          0          0        0
inet.0           0          0          0          0          0        0
Peer                AS     InPkt     OutPkt  OutQ   Flaps Last Up/Dwn State
192.0.2.1       65000        44        46      0       0       18:27 Establ
  bgp.l2vpn.0: 2/2/2/0
192.0.2.2       65000        43        45      0       0       18:22 Establ
  bgp.l2vpn.0: 2/2/2/0
```

```
192.0.2.3       65000        42        45      0       0       18:19 Establ
  bgp.l2vpn.0: 1/1/1/0
192.0.2.4       65000        43        45      0       0       18:15 Establ
  bgp.l2vpn.0: 1/1/1/0
```

6. In NG-VPLS, point-to-multipoint LSPs carry only unknown unicast, broadcast, and multicast packets. A full mesh of point-to-point LSPs is needed between the PE routers for NG-VPLS. The point-to-point LSPs create routes in the `inet.3` routing table. These entries are used to resolve the Layer 2 VPN routes received from the BGP peers. All other data traffic is sent over point-to-point LSPs.

   A point-to-point LSP is also created for the route reflector. This LSP creates a route in the `inet.3` routing table for BGP next-hop resolution.

   On Router PE1, use the `show mpls lsp` command to verify that the `to-PE2`, `to-PE3`, `to-PE4`, and `to-RR` LSPs are in the `Up` state.

```
user@PE1> show mpls lsp ingress unidirectional
Ingress LSP: 7 sessions
To              From            State Rt P     ActivePath       LSPname
192.0.2.2         192.0.2.1         Up     0 *                      to-PE2
192.0.2.3         192.0.2.1         Up     0 *                      to-PE3
192.0.2.4         192.0.2.1         Up     0 *                      to-PE4
192.0.2.7         192.0.2.1         Up     0 *                      to-RR
Total 4 displayed, Up 4, Down 0
admin@PE2# run show mpls lsp ingress unidirectional
Ingress LSP: 7 sessions
To              From            State Rt P     ActivePath       LSPname
192.0.2.1         192.0.2.2         Up     0 *                      to-PE1
192.0.2.3         192.0.2.2         Up     0 *                      to-PE3
192.0.2.4         192.0.2.2         Up     0 *                      to-PE4
192.0.2.7         192.0.2.2         Up     0 *                      to-RR
Total 4 displayed, Up 4, Down 0
admin@PE3# run show mpls lsp ingress unidirectional
Ingress LSP: 7 sessions
To              From            State Rt P     ActivePath       LSPname
192.0.2.1         192.0.2.3         Up     0 *                      to-PE1
192.0.2.2         192.0.2.3         Up     0 *                      to-PE2
192.0.2.4         192.0.2.3         Up     0 *                      to-PE4
192.0.2.7         192.0.2.3         Up     0 *                      to-RR
Total 4 displayed, Up 4, Down 0
admin@PE4# run show mpls lsp ingress unidirectional
```

```
Ingress LSP: 7 sessions
To              From            State Rt P     ActivePath       LSPname
192.0.2.1         192.0.2.4        Up     0 *                     to-PE1
192.0.2.2         192.0.2.4        Up     0 *                     to-PE2
192.0.2.3         192.0.2.4        Up     0 *                     to-PE3
192.0.2.7         192.0.2.4        Up     0 *                     to-RR
Total 4 displayed, Up 4, Down 0
```

7. For each VPLS instance, a PE router creates a dedicated point-to-multipoint LSP. In this example, Router PE1 and Router PE2 are configured to use a point-to-multipoint dynamic template.

   For dynamic point-to-multipoint LSPs, whenever VPLS discovers a new Layer 2 VPN neighbor through BGP, a source-to-leaf sub-LSP is added in the VPLS instance for this neighbor PE router.

   On Router PE1, use the `show mpls lsp` command to verify that three source-to-leaf sub-LSPs are created.

```
user@PE1> show mpls lsp ingress p2mp
Ingress LSP: 1 sessions
P2MP name: 192.0.2.1:1:vpls:GOLD, P2MP branch count: 3
To            From          State Rt P   ActivePath     LSPname
192.0.2.4       192.0.2.1       Up     0 *       192.0.2.4:192.0.2.1:1:vpls:GOLD
192.0.2.3       192.0.2.1       Up     0 *       192.0.2.3:192.0.2.1:1:vpls:GOLD
192.0.2.2       192.0.2.1       Up     0 *       192.0.2.2:192.0.2.1:1:vpls:GOLD
Total 3 displayed, Up 3, Down 0
```

8. On Router PE2, use the `show mpls lsp` command to verify that three source-to-leaf sub-LSPs are created.

```
user@PE2> show mpls lsp p2mp ingress
Ingress LSP: 1 sessions
P2MP name: 192.0.2.2:10:vpls:GOLD, P2MP branch count: 3
To            From          State Rt P     ActivePath       LSPname
192.0.2.4       192.0.2.2       Up     0 *        192.0.2.4:192.0.2.2:10:vpls:GOLD
192.0.2.3       192.0.2.2       Up     0 *        192.0.2.3:192.0.2.2:10:vpls:GOLD
192.0.2.1       192.0.2.2       Up     0 *        192.0.2.1:192.0.2.2:10:vpls:GOLD
Total 3 displayed, Up 3, Down 0
```

9. In this step, Router PE3 and Router PE4 are using static point-to-multipoint LSPs. For static point-to-multipoint LSPs, the source-to-leaf sub-LSPs to all the PE routers are manually configured.

On Router PE3, use the `show mpls lsp` command to verify that three source-to-leaf sub-LSPs have been configured.

```
user@PE3> show mpls lsp p2mp ingress
Ingress LSP: 1 sessions
P2MP name: vpls-GOLD, P2MP branch count: 3
To             From          State Rt P     ActivePath       LSPname
192.0.2.1       192.0.2.3       Up     0 *                       to-pe1
192.0.2.4       192.0.2.3       Up     0 *                       to-pe4
192.0.2.2       192.0.2.3       Up     0 *                       to-pe2
Total 3 displayed, Up 3, Down 0
```

10. On Router PE4, use the `show mpls lsp` command to verify that three source-to-leaf sub-LSPs are configured.

```
user@PE4> show mpls lsp ingress p2mp
Ingress LSP: 1 sessions
P2MP name: vpls-GOLD, P2MP branch count: 3
To             From          State Rt P     ActivePath       LSPname
192.0.2.1       192.0.2.4       Up     0 *                       to-pe1
192.0.2.3       192.0.2.4       Up     0 *                       to-pe3
192.0.2.2       192.0.2.4       Up     0 *                       to-pe2
Total 3 displayed, Up 3, Down 0
```

11. Each point-to-multipoint LSP created by the PE router can be identified using an RSVP-TE point-to-multipoint session object. The session object is passed as a PMSI tunnel attribute by BGP when it advertises VPLS routes. Using this tunnel attribute, an incoming source-to-leaf sub LSP add request (RSVP-Path message) supports label allocation in such a way that when traffic arrives on this source-to-leaf sub-LSP the router terminates the message in the right VPLS instance and also identifies the originating PE. This supports source MAC address learning.

On Router PE1, use the `show rsvp session` command to verify that the RSVP session for the dynamic point-to-multipoint LSP is `Up` and that link protection is configured as `desired`. Notice that the point-to-multipoint session object to be sent in BGP is `54337`.

```
user@PE1> show rsvp session detail p2mp ingress
Ingress RSVP: 7 sessions
P2MP name: 192.0.2.1:1:vpls:GOLD, P2MP branch count: 3

192.0.2.2
```

```
  From: 192.0.2.1, LSPstate: Up, ActiveRoute: 0
  LSPname: 192.0.2.2:192.0.2.1:1:vpls:GOLD, LSPpath: Primary
  P2MP LSPname: 192.0.2.1:1:vpls:GOLD
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 262145
  Resv style: 1 SE, Label in: -, Label out: 262145
  Time left:    -, Since: Tue Oct  6 16:27:23 2009
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 2 receiver 54337 protocol 0
  Link protection desired
  Type: Protection down
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 10.10.2.2 (xe-0/1/0.0) 371 pkts
  RESV rcvfrom: 10.10.2.2 (xe-0/1/0.0) 370 pkts
  Explct route: 10.10.2.2
  Record route: <self> 10.10.2.2
```

12. Router PE4 is configured for static point-to-multipoint LSPs. Link protection is not configured for these LSPs. Use the `show rsvp session` command to verify that the point-to-multipoint session object to be sent in BGP is `42873`.

```
user@PE4> show rsvp session detail p2mp ingress
Ingress RSVP: 7 sessions
P2MP name: vpls-GOLD, P2MP branch count: 3

192.0.2.1
  From: 192.0.2.4, LSPstate: Up, ActiveRoute: 0
  LSPname: to-pe1, LSPpath: Primary
  P2MP LSPname: vpls-GOLD
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 390416
  Resv style: 1 SE, Label in: -, Label out: 390416
  Time left:    -, Since: Tue Oct  6 15:28:33 2009
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 10 receiver 42873 protocol 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 10.10.9.1 (xe-0/0/0.0) 524 pkts
  RESV rcvfrom: 10.10.9.1 (xe-0/0/0.0) 447 pkts
  Explct route: 10.10.9.1 10.10.3.1
  Record route: <self> 10.10.9.1 10.10.3.1
```

13. On Router PE1, use the `show route table` command to verify that Router PE1 received a Layer 2 VPN route to Router PE2 from the router reflector and the route includes a PMSI object that contains the point-to-multipoint tunnel identifier of `20361`.

```
user@PE1> show route table GOLD.l2vpn.0 detail
GOLD.l2vpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
!
!
192.0.2.2:10:3:1/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.0.2.2:10
                PMSI: Flags 0:RSVP-TE:label[0:0:0]:Session_13[192.0.2.2:0:20361:192.0.2.2]
                Next hop type: Indirect
                Next-hop reference count: 7
                Source: 192.0.2.7
                Protocol next hop: 192.0.2.2
                Indirect next hop: 2 no-forward
                State: <Secondary Active Int Ext>
                Local AS: 65000 Peer AS: 65000
                Age: 4:25:25    Metric2: 1
                Task: BGP_65000.192.0.2.7+63544
                Announcement bits (1): 0-GOLD-l2vpn
                AS path: I (Originator) Cluster list:  192.0.2.7
                AS path:  Originator ID: 192.0.2.2
                Communities: target:65000:1 Layer2-info: encaps:VPLS, control flags:, mtu:
0, site preference: 100
                Import Accepted
                Label-base: 262145, range: 8
                Localpref: 100
                Router ID: 192.0.2.7
                Primary Routing Table bgp.l2vpn.0
PMSI: Flags 0:RSVP-TE:label[0:0:0]:Session_13[192.0.2.2:0:20361:192.0.2.2]
```

14. On Router PE2, use the `show rsvp session` command to verify that the PMSI tunnel identifier object of `20361` matches the PMSI tunnel identifier object displayed on Router PE1.

```
user@PE2> show rsvp session p2mp detail
Ingress RSVP: 7 sessions
```

```
P2MP name: 192.0.2.2:10:vpls:GOLD, P2MP branch count: 3

192.0.2.1
  From: 192.0.2.2, LSPstate: Up, ActiveRoute: 0
  LSPname: 192.0.2.1:192.0.2.2:10:vpls:GOLD, LSPpath: Primary
  P2MP LSPname: 192.0.2.2:10:vpls:GOLD
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 262171
  Resv style: 1 SE, Label in: -, Label out: 262171
  Time left:    -, Since: Tue Oct  6 16:31:47 2009
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 20361 protocol 0
  Link protection desired
  Type: Protection down
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 10.10.2.1 (xe-0/1/0.0) 379 pkts
  RESV rcvfrom: 10.10.2.1 (xe-0/1/0.0) 379 pkts
  Explct route: 10.10.2.1
  Record route: <self> 10.10.2.1
```

Verifying the Data Plane

Step-by-Step Procedure

After the control plane is verified using the previous steps, you can verify the data plane. This section describes `show` command outputs you can use to validate the data plane.

1. On Router PE1, use the `show vpls connections extensive | match Flood` command to verify the point-to-multipoint LSP name and status of all the sites. Notice the flood next-hop identifier of `600` for the `192.0.2.1:1:vpls:GOLD` LSP.

```
user@PE1> show vpls connections extensive | match Flood
Ingress RSVP-TE P2MP LSP: 192.0.2.1:1:vpls:GOLD, Flood next-hop ID: 600
```

2. On Router PE1, use the `show vpls connections extensive` command to verify the point-to-multipoint LSP name and status of all the sites.

```
user@PE1> show vpls connections extensive
Instance: GOLD
  Local site: CE1 (1)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    IRB interface present: no
    ge-1/0/0.1
    lsi.1049353     3       Intf - vpls GOLD local site 1 remote site 3
    lsi.1049346     4       Intf - vpls GOLD local site 1 remote site 4
        Interface flags: VC-Down
    lsi.1049354     5       Intf - vpls GOLD local site 1 remote site 5
    lsi.1049355     6       Intf - vpls GOLD local site 1 remote site 6
    Label-base        Offset     Range     Preference
    262169            1          8         100
    connection-site           Type  St     Time last up      # Up trans
    3                         rmt   Up     Oct  6 16:27:23 2009       1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 262171, Outgoing label: 262145
      Local interface: lsi.1049353, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 3
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.2:192.0.2.1:1:vpls:GOLD, State: Up
        Egress branch LSP:  192.0.2.1:192.0.2.2:10:vpls:GOLD, State: Up
    Connection History:
        Oct  6 16:27:23 2009  status update timer
        Oct  6 16:27:23 2009  PE route changed
        Oct  6 16:27:23 2009  Out lbl Update                    262145
        Oct  6 16:27:23 2009  In lbl Update                     262171
        Oct  6 16:27:23 2009  loc intf up                  lsi.1049353
    4                         rmt   RM
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.2:192.0.2.1:1:vpls:GOLD, State: Up
    5                         rmt   Up     Oct  6 16:27:27 2009          1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 262173, Outgoing label: 262145
      Local interface: lsi.1049354, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 5
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.3:192.0.2.1:1:vpls:GOLD, State: Up
        Egress branch LSP:  to-pe1, State: Up
    Connection History:
        Oct  6 16:27:27 2009  status update timer
        Oct  6 16:27:27 2009  PE route changed
        Oct  6 16:27:27 2009  Out lbl Update                    262145
        Oct  6 16:27:27 2009  In lbl Update                     262173
        Oct  6 16:27:27 2009  loc intf up                  lsi.1049354
    6                         rmt   Up     Oct  6 16:27:31 2009         1
      Remote PE: 192.0.2.4, Negotiated control-word: No
      Incoming label: 262174, Outgoing label: 800000
      Local interface: lsi.1049355, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls GOLD local site 1 remote site 6
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.4:192.0.2.1:1:vpls:GOLD, State: Up
        Egress branch LSP:  to-pe1, State: Up
    Connection History:
        Oct  6 16:27:31 2009  status update timer
        Oct  6 16:27:31 2009  PE route changed
        Oct  6 16:27:31 2009  Out lbl Update                    800000
        Oct  6 16:27:31 2009  In lbl Update                     262174
        Oct  6 16:27:31 2009  loc intf up                  lsi.1049355
  Local site: Direct (2)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    IRB interface present: no
  Interface name  Remote site ID  Description
    ge-1/1/0.1
    lsi.1049347     3       Intf - vpls GOLD local site 2 remote site 3
        Interface flags: VC-Down
    lsi.1049348     4       Intf - vpls GOLD local site 2 remote site 4
        Interface flags: VC-Down
    lsi.1049350     5       Intf - vpls GOLD local site 2 remote site 5
        Interface flags: VC-Down
    lsi.1049352     6       Intf - vpls GOLD local site 2 remote site 6
        Interface flags: VC-Down
    Label-base        Offset     Range     Preference
    262177            1          8         100
    connection-site           Type  St     Time last up
    3                         rmt   LM
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.2:192.0.2.1:1:vpls:GOLD, State: Up
    4                         rmt   LM
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.2:192.0.2.1:1:vpls:GOLD, State: Up
    5                        rmt   LM
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.3:192.0.2.1:1:vpls:GOLD, State: Up
    6                        rmt   LM
      RSVP-TE P2MP lsp:
        Ingress branch LSP: 192.0.2.4:192.0.2.1:1:vpls:GOLD, State: Up
  Ingress RSVP-TE P2MP LSP: 192.0.2.1:1:vpls:GOLD, Flood next-hop ID: 600
```

3. Junos OS Release 9.0 and later identifies the flood next-hop route as a composite next hop. On Router PE1, use the `show route forwarding-table family vpls vpn GOLD detail` command to verify that three composite flood next-hop routes are installed in the Packet Forwarding Engine.

```
user@PE1> show route forwarding-table family vpls vpn GOLD detail
Routing table: GOLD.vpls
VPLS:
Destination        Type RtRef Next hop       Type Index NhRef Netif
default            perm     0                dscd   518     1
00:00:28:28:28:02/48
                   user     0                ucst   617     4 ge-1/1/0.1
00:00:28:28:28:06/48
                   user     0                indr 1048576     4
                              10.10.3.2     Push 800000, Push 390384(top)   621     2
xe-0/2/0.0
lsi.1049353        intf     0                indr 1048574     3
                              10.10.2.2     Push 262145   598     2 xe-0/1/0.0
lsi.1049354        intf     0                indr 1048575     4
                              10.10.1.2     Push 262145, Push 302272(top)   602     2
xe-0/3/0.0
lsi.1049355        intf     0                indr 1048576     4
                              10.10.3.2     Push 800000, Push 390384(top)   621     2
xe-0/2/0.0
00:14:f6:75:78:00/48
                   user     0                indr 1048575     4
                              10.10.1.2     Push 262145, Push 302272(top)   602     2
xe-0/3/0.0
00:19:e2:57:e7:c0/48
                   user     0                ucst   604     4 ge-1/0/0.1
0x30003/51         user     0                comp   613     2
0x30002/51         user     0                comp   615     2
0x30001/51         user     0                comp   582     2
```

```
ge-1/0/0.1          intf     0                 ucst   604      4 ge-1/0/0.1
ge-1/1/0.1          intf     0                 ucst   617      4 ge-1/1/0.1
```

You can also use the use the `show route forwarding-table family vpls extensive` command to match the flood identifier and note the flood label. To match the label out corresponding to the point-to-multipoint LSP, use the `show rsvp session ingress p2mp` command.

4. On Router PE1, use the `show route forwarding-table family vpls vpn GOLD extensive | find 0x30003/51` command to get more details about the composite next-hop route and the associated point-to-multipoint LSP labels.

```
user@PE1> show route forwarding-table family vpls vpn GOLD extensive | find 0x30003/51
Destination:  0x30003/51
  Route type: user
  Route reference: 0                   Route interface-index: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 613      Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 556      Reference: 4
  Next-hop type: unicast               Index: 604      Reference: 4
  Next-hop interface: ge-1/0/0.1
  Next-hop type: unicast               Index: 617      Reference: 4
  Next-hop interface: ge-1/1/0.1

Destination:  0x30002/51
  Route type: user
  Route reference: 0                   Route interface-index: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 615      Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 556      Reference: 4
  Next-hop type: unicast               Index: 604      Reference: 4
  Next-hop interface: ge-1/0/0.1
  Next-hop type: unicast               Index: 617      Reference: 4
  Next-hop interface: ge-1/1/0.1
  Nexthop:
  Next-hop type: composite             Index: 603      Reference: 3
  Next-hop type: flood                 Index: 600      Reference: 2
  Nexthop: 10.10.2.2
  Next-hop type: Push 262145           Index: 599      Reference: 1
  Next-hop interface: xe-0/1/0.0
  Nexthop: 10.10.3.2
  Next-hop type: Push 390496          Index: 622     Reference: 1
  Next-hop interface: xe-0/2/0.0
  Nexthop: 10.10.1.2
  Next-hop type: Push 302416          Index: 618     Reference: 1
  Next-hop interface: xe-0/3/0.0

Destination:  0x30001/51
  Route type: user
  Route reference: 0                  Route interface-index: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite            Index: 582     Reference: 2
  Nexthop:
  Next-hop type: composite            Index: 556     Reference: 4
  Next-hop type: unicast              Index: 604     Reference: 4
  Next-hop interface: ge-1/0/0.1
  Next-hop type: unicast              Index: 617     Reference: 4
  Next-hop interface: ge-1/1/0.1
  Nexthop:
  Next-hop type: composite            Index: 603     Reference: 3
  Next-hop type: flood                Index: 600     Reference: 2
  Nexthop: 10.10.2.2
  Next-hop type: Push 262145          Index: 599     Reference: 1
  Next-hop interface: xe-0/1/0.0
  Nexthop: 10.10.3.2
  Next-hop type: Push 390496          Index: 622     Reference: 1
  Next-hop interface: xe-0/2/0.0
  Nexthop: 10.10.1.2
  Next-hop type: Push 302416          Index: 618     Reference: 1
  Next-hop interface: xe-0/3/0.0

Destination:  ge-1/0/0.1
  Route type: interface
  Route reference: 0                  Route interface-index: 84
  Flags: sent to PFE
  Next-hop type: unicast              Index: 604     Reference: 4
  Next-hop interface: ge-1/0/0.1

Destination:  ge-1/1/0.1
  Route type: interface
  Route reference: 0                  Route interface-index: 86
```

```
    Flags: sent to PFE
    Next-hop type: unicast              Index: 617      Reference: 4
    Next-hop interface: ge-1/1/0.1
```

5. On Router PE1, use the `show vpls mac-table instance GOLD` command to verify the learned MAC addresses of CE routers connected to the VPLS domain.

```
user@PE1> show vpls mac-table instance GOLD
MAC flags (S -static MAC, D -dynamic MAC,
           SE -Statistics enabled, NM -Non configured MAC)

Routing instance : GOLD
 Bridging domain : __GOLD__, VLAN : NA
   MAC                  MAC      Logical
   address              flags    interface
   00:00:28:28:28:02    D        ge-1/1/0.1
   00:00:28:28:28:04    D        lsi.1049353
   00:14:f6:75:78:00    D        lsi.1049354
   00:19:e2:51:7f:c0    D        lsi.1049353
   00:19:e2:57:e7:c0    D        ge-1/0/0.1
```

6. On Router PE1, use the `show vpls statistics` command to verify the broadcast, multicast, and unicast traffic flow using the packet statistics for the VPLS instance.

```
user@PE1> show vpls statistics
VPLS statistics:

Instance: GOLD
   Local interface: lsi.1049347, Index: 72
     Current MAC count:                     0
   Local interface: lsi.1049348, Index: 73
     Current MAC count:                     0
   Local interface: lsi.1049346, Index: 82
     Current MAC count:                     0
   Local interface: lsi.1049353, Index: 83
   Remote PE: 192.0.2.2
     Current MAC count:                     2
   Local interface: ge-1/0/0.1, Index: 84
     Broadcast packets:                   421
     Broadcast bytes  :                 26944
     Multicast packets:                  3520
```

```
    Multicast bytes  :               261906
    Flooded packets  :            509043345
    Flooded bytes    :         130315095486
    Unicast packets  :            393836428
    Unicast bytes    :         100822118854
    Current MAC count:                    1 (Limit 1024)
  Local interface: ge-1/1/0.1, Index: 86
    Broadcast packets:                    0
    Broadcast bytes  :                    0
    Multicast packets:                    0
    Multicast bytes  :                    0
    Flooded packets  :             22889544
    Flooded bytes    :           5859702144
    Unicast packets  :                  472
    Unicast bytes    :                30838
    Current MAC count:                    1 (Limit 1024)
  Local interface: lsi.1049354, Index: 88
  Remote PE: 192.0.2.3
    Current MAC count:                    1
  Local interface: lsi.1049350, Index: 89
    Current MAC count:                    0
  Local interface: lsi.1049355, Index: 90
  Remote PE: 192.0.2.4
    Current MAC count:                    0
  Local interface: lsi.1049352, Index: 91
    Current MAC count:                    0
```

Results

The configuration, verification, and testing part of this example has been completed. The following section is for your reference.

The relevant sample configuration for Router PE1 follows.

PE1 Configuration

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ethernet;
}
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.1/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.3.1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/0 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    ge-1/1/0 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.1/24;
            }
        }
    }
}
routing-options {
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path vpls-GOLD-p2mp-template {
            template;
            optimize-timer 50;
            link-protection;
            p2mp;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group to-RR {
            type internal;
            local-address 192.0.2.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/0/0.1;
        interface ge-1/1/0.1;
        route-distinguisher 192.0.2.1:1;
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    vpls-GOLD-p2mp-template;
                }
            }
        }
        vrf-target target:65000:1;
        protocols {
            vpls {
                site-range 8;
                no-tunnel-services;
                site CE1 {
                    site-identifier 1;
                    interface ge-1/0/0.1;
                }
                site Direct {
                    site-identifier 2;
                    interface ge-1/1/0.1;
                }
            }
        }
    }
}
```

The relevant sample configuration for Router PE2 follows.

PE2 Configuration

```
chassis {
    dump-on-panic;
    aggregated-devices {
        ethernet {
            device-count 1;
        }
    }
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
}
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.2/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.4.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/1 {
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-1/0/2 {
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-1/1/0 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    ae0 {
        vlan-tagging;
        encapsulation vlan-vpls;
        unit 1 {
            encapsulation vlan-vpls;
            vlan-id 1000;
            family vpls;
        }
    }
    fxp0 {
        apply-groups [ re0 re1 ];
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.2/24;
            }
        }
    }
}
routing-options {
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path vpls-GOLD-p2mp-template {
            template;
            optimize-timer 50;
            link-protection;
            p2mp;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group to-RR {
            type internal;
            local-address 192.0.2.2;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
routing-instances {
    GOLD {
        instance-type vpls;
        interface ge-1/1/0.1;
        interface ae0.1;
        route-distinguisher 192.0.2.2:10;
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    vpls-GOLD-p2mp-template;
                }
            }
        }
        vrf-target target:65000:1;
        protocols {
            vpls {
                site-range 8;
                site CE1 {
                    site-identifier 3;
                    interface ae0.1;
                }
                site Direct {
                    site-identifier 4;
                    interface ge-1/1/0.1;
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

Next-Generation VPLS Point-to-Multipoint Forwarding Overview | 1028

Flooding Unknown Traffic Using Point-to-Multipoint LSPs in VPLS

IN THIS SECTION

- Configuring Static Point-to-Multipoint Flooding LSPs | **1078**
- Configuring Dynamic Point-to-Multipoint Flooding LSPs | **1079**

For a VPLS routing instance, you can flood unknown unicast, broadcast, and multicast traffic using point-to-multipoint (also called *P2MP*) LSPs. By default, VPLS relies upon ingress replication to flood unknown traffic to the members of a VPLS routing instance. This can cause replication of data at routing nodes shared by multiple VPLS members, as shown in Figure 80 on page 1077. The flood data is tripled between PE router PE1 and provider router P1 and doubled between provider routers P1 and P2. By configuring point-to-multipoint LSPs to handle flood traffic, the VPLS routing instance can avoid this type of traffic replication in the network, as shown in Figure 81 on page 1077.

Figure 80: Flooding Unknown VPLS Traffic Using Ingress Replication

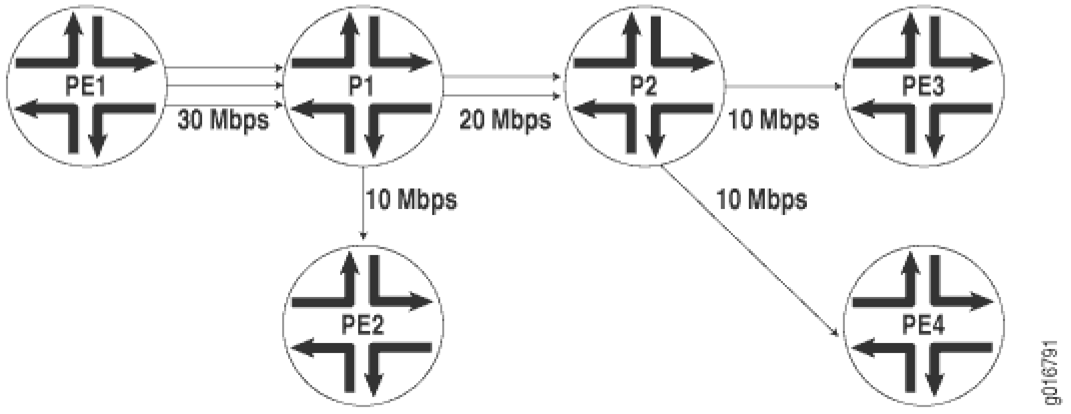
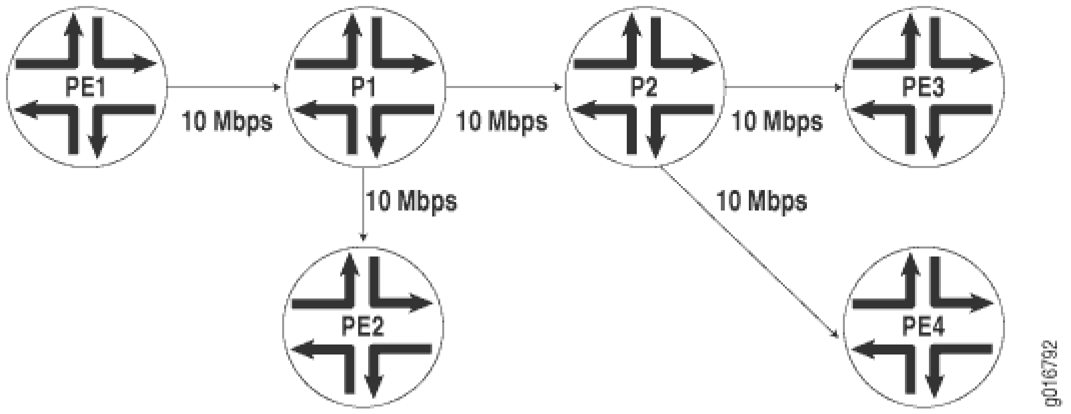


Figure 81: Flooding Unknown VPLS Traffic Using a Point-to-Multipoint LSP



NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

The point-to-multipoint LSP used for VPLS flooding can be either static or dynamic. In either case, for each VPLS routing instance, the PE router creates a dedicated point-to-multipoint LSP. All of the neighbors of the VPLS routing instance are added to the point-to-multipoint LSP when the feature is enabled. If there are *n* PE routers in the VPLS routing instance, *n* point-to-multipoint LSPs are created in the network where each PE router is the root of the point-to-multipoint tree and includes the rest of the *n – 1* PE routers as leaf nodes. If you configured static point-to-multipoint LSPs for flooding, any additional VPLS neighbors added to the routing instance later are not automatically added to the point-to-multipoint LSP. You will need to manually add the new VPLS neighbors to the static point-to-multipoint flooding LSP. If you configure dynamic point-to-multipoint LSPs, whenever VPLS discovers a new neighbor through BGP, a sub-LSP for this neighbor is added to the point-to-multipoint LSP for the routing instance.

This feature can be enabled incrementally on any PE router that is part of a specific VPLS routing instance. The PE routers can then use point-to-multipoint LSPs to flood traffic, whereas other PE routers in the same VPLS routing instance can still use ingress replication to flood traffic. However, when this feature is enabled on any PE router, you must ensure that all PE routers in the VPLS routing instance that participate in the flooding of traffic over point-to-multipoint LSPs are upgraded to Junos OS Release 8.3 or later to support this feature.

To flood unknown unicast, broadcast, and multicast traffic using point-to-multipoint LSPs, configure the `rsvp-te` statement as follows:

```
rsvp-te {
    label-switched-path-template (Multicast) {
        (default-template | lsp-template-name);
    }
    static-lsp lsp-name;
}
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instance routing-instance-name provider-tunnel]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name provider-tunnel]`

You can configure either a static point-to-multipoint LSP for VPLS flooding or a dynamic point-to-multipoint LSP.

NOTE: You cannot specify both the `static` and `label-switched-path-template` statements at the same time.

The following sections describe how to configure static and dynamic point-to-multipoint LSPs for flooding unknown traffic in a VPLS routing instance:

Configuring Static Point-to-Multipoint Flooding LSPs

The `static-lsp` option creates a static flooding point-to-multipoint LSP that includes all of the neighbors in the VPLS routing instance. Flood traffic is sent to all of the VPLS neighbors using the generated point-to-multipoint LSP. VPLS neighbors added to the routing instance later are not automatically added to the point-to-multipoint LSP. You will need to manually add the new VPLS neighbors to the static point-to-multipoint flooding LSP. By configuring static point-to-multipoint LSPs for flooding, you have more control over which path each sub-LSP follows.

To configure a static flooding point-to-multipoint LSP, specify the name of the static flooding point-to-multipoint LSP by including the `static-lsp` statement:

```
static-lsp lsp-name;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name provider-tunnel rsvp-te]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name provider-tunnel rsvp-te]`

Configuring Dynamic Point-to-Multipoint Flooding LSPs

To configure a dynamic point-to-multipoint flooding LSP, include the `label-switched-path-template` statement option at the `[edit routing-instances routing-instance-name provider-tunnel rsvp-te]` hierarchy level:

```
[edit routing-instances routing-instance-name provider-tunnel rsvp-te]
label-switched-path-template (Multicast) {
    (default-template | lsp-template-name);
}
```

You can automatically generate the point-to-multipoint LSP to be used for flooding unknown traffic or you can manually configure the point-to-multipoint LSP:

Configuring Dynamic Point-to-Multipoint Flooding LSPs with the Default Template

The `default-template` option, specified at the `[edit routing-instances routing-instance-name provider-tunnel rsvp-te label-switched-path-template]` hierarchy level, causes the point-to-multipoint LSPs to be created with default parameters. The default parameters are for a minimally configured point-to-multipoint LSP. The name of this point-to-multipoint LSP is also generated automatically and is based on the following model:

id:vpls:*router-id*:*routing-instance-name*

The following `show` command output for `show mpls lsp p2mp` illustrates how a point-to-multipoint flood LSP name could appear if you configure the `label-switched-path-template` statement with the `default-template` option:

```
user@host> show mpls lsp p2mp ingress
Ingress LSP: 2 sessions P2MP name: static, P2MP branch count: 3
To              From            State Rt ActivePath       P     LSPname
```

```
10.255.14.181   10.255.14.172   Up     0                 *     vpn02-vpn11
10.255.14.177   10.255.14.172   Up     0 path2           *     vpn02-vpn07
10.255.14.174   10.255.14.172   Up     0 path3           *     vpn02-vpn04
P2MP name: 9:vpls:10.255.14.172:green, P2MP branch count: 2
To              From            State Rt ActivePath      P     LSPname
10.255.14.177   10.255.14.172   Up     0                 *     11:vpls:10.255.14.172:green
10.255.14.174   10.255.14.172   Up     0                 *     10:vpls:10.255.14.172:green
Total 5 displayed, Up 5, Down 0
```

The dynamically generated point-to-multipoint LSP name is `9:vpls:10.255.14.172:green`.

Configuring Dynamic Point-to-Multipoint Flooding LSPs with a Preconfigured Template

You can configure a point-to-multipoint flooding LSP template for the VPLS routing instance. The template allows you to specify the properties of the dynamic point-to-multipoint LSPs that are used to flood traffic for the VPLS routing instance. You can specify all of the standard options available for a point-to-multipoint LSP within this template. These properties are inherited by the dynamic point-to-multipoint flood LSPs.

To configure a point-to-multipoint LSP template for flooding VPLS traffic, specify all of the properties you want to include in a point-to-multipoint LSP configuration. To specify this LSP as a point-to-multipoint flooding template, include the `p2mp` and `template` statements:

```
p2mp;
template;
```

You can include these statements at the following hierarchy levels:

- `[edit protocols mpls label-switched-path` *`p2mp-lsp-template-name`*`]`
- `[edit logical-systems` *`logical-system-name`* `protocols mpls label-switched-path` *`p2mp-lsp-template-name`*`]`

For more information about how to configure the `p2mp` statement and point-to-multipoint LSPs, see the MPLS Applications User Guide.

Once you have configured the point-to-multipoint LSP template, specify the name of the point-to-multipoint LSP template with the `label-switched-path-template` statement:

```
label-switched-path-template (Multicast) p2mp-lsp-template-name;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `provider-tunnel rsvp-te]`

- `[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* provider-tunnel rsvp-te]`

Example: Configuring Ingress Replication for IP Multicast Using MBGP MVPNs

IN THIS SECTION

- Requirements | **1081**
- Overview | **1081**
- Configuration | **1084**
- Verification | **1089**

Requirements

The routers used in this example are Juniper Networks M Series Multiservice Edge Routers, T Series Core Routers, or MX Series 5G Universal Routing Platforms. When using ingress replication for IP multicast, each participating router must be configured with BGP for control plane procedures and with ingress replication for the data provider tunnel, which forms a full mesh of MPLS point-to-point LSPs. The ingress replication tunnel can be selective or inclusive, depending on the configuration of the provider tunnel in the routing instance.

Overview

IN THIS SECTION

- Topology | **1082**

The `ingress-replication` provider tunnel type uses unicast tunnels between routers to create a multicast distribution tree.

The `mpls-internet-multicast` routing instance type uses ingress replication provider tunnels to carry IP multicast data between routers through an MPLS cloud, using MBGP (or Next Gen) MVPN. Ingress replication can also be configured when using MVPN to carry multicast data between PE routers.

The `mpls-internet-multicast` routing instance is a non-forwarding instance used only for control plane procedures. It does not support any interface configurations. Only one `mpls-internet-multicast` routing instance can be defined for a logical system. All multicast and unicast routes used for IP multicast are associated only with the default routing instance (`inet.0`), not with a configured routing instance. The `mpls-internet-multicast` routing instance type is configured for the default master instance on each router, and is also included at the `[edit protocols pim]` hierarchy level in the default instance.

For each `mpls-internet-multicast` routing instance, the `ingress-replication` statement is required under the `provider-tunnel` statement and also under the `[edit routing-instances` **`routing-instance-name`** `provider-tunnel selective group source]` hierarchy level.

When a new destination needs to be added to the ingress replication provider tunnel, the resulting behavior differs depending on how the ingress replication provider tunnel is configured:

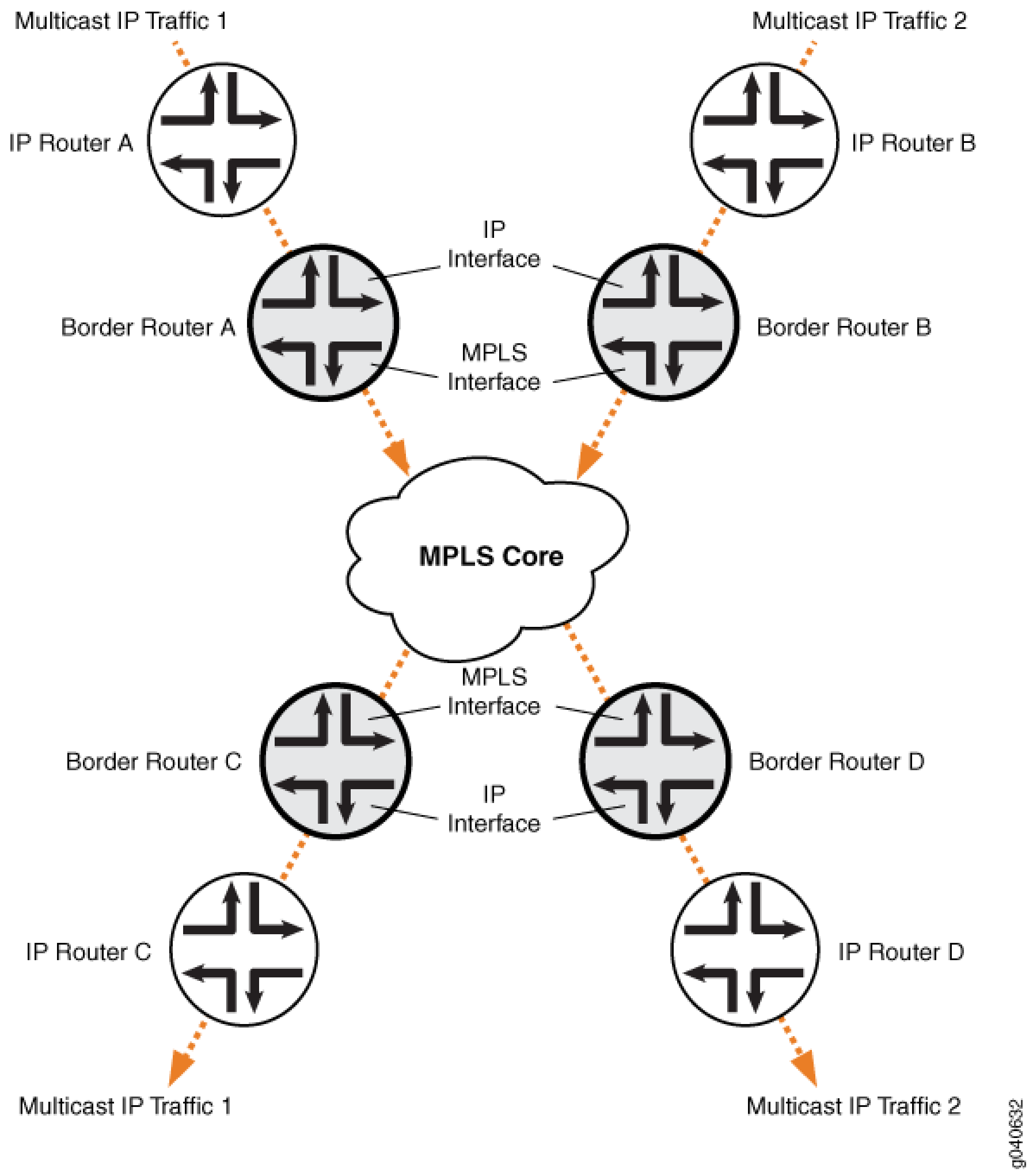
- `create-new-ucast-tunnel`—When this statement is configured, a new unicast tunnel to the destination is created, and is deleted when the destination is no longer needed. Use this mode for RSVP LSPs using ingress replication.

- `label-switched-path-template (Multicast)`—When this statement is configured, an LSP template is used for the for the point-to-multipoint LSP for ingress replication.

Topology

The IP topology consists of routers on the edge of the IP multicast domain. Each router has a set of IP interfaces configured toward the MPLS cloud and a set of interfaces configured toward the IP routers. See Figure 82 on page 1083. Internet multicast traffic is carried between the IP routers, through the MPLS cloud, using ingress replication tunnels for the data plane and a full-mesh IBGP session for the control plane.

Figure 82: Internet Multicast Topology



Configuration

IN THIS SECTION

● Procedure | 1084

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.

Border Router C

```
set protocols mpls ipv6-tunneling
set protocols mpls interface all
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.10.61
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn any
set protocols bgp group ibgp family inet6 unicast
set protocols bgp group ibgp family inet6-vpn any
set protocols bgp group ibgp family inet-mvpn signaling
set protocols bgp group ibgp family inet6-mvpn signaling
set protocols bgp group ibgp export to-bgp
set protocols bgp group ibgp neighbor 10.255.10.97
set protocols bgp group ibgp neighbor 10.255.10.55
set protocols bgp group ibgp neighbor 10.255.10.57
set protocols bgp group ibgp neighbor 10.255.10.59
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface so-1/3/1.0
set protocols ospf area 0.0.0.0 interface so-0/3/0.0
set protocols ospf3 area 0.0.0.0 interface lo0.0
set protocols ospf3 area 0.0.0.0 interface so-1/3/1.0
set protocols ospf3 area 0.0.0.0 interface so-0/3/0.0
set protocols ldp interface all
set protocols pim rp static address 192.0.2.2
set protocols pim rp static address 2::192.0.2.2
set protocols pim interface fe-0/1/0.0
set protocols pim mpls-internet-multicast
set routing-instances test instance-type mpls-internet-multicast
set routing-instances test provider-tunnel ingress-replication label-switched-path
set routing-instances test protocols mvpn
```

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.

The following example shows how to configure ingress replication on an IP multicast instance with the routing instance type `mpls-internet-multicast`. Additionally, this example shows how to configure a selective provider tunnel that selects a new unicast tunnel each time a new destination needs to be added to the multicast distribution tree.

This example shows the configuration of the link between Border Router C and edge IP Router C, from which Border Router C receives PIM join messages.

1. Enable MPLS.

   ```
   [edit protocols mpls]
   user@Border_Router_C# set ipv6-tunneling
   user@Border_Router_C# set interface all
   ```

2. Configure a signaling protocol, such as RSVP or LDP.

   ```
   [edit protocols ldp]
   user@Border_Router_C# set interface all
   ```

3. Configure a full-mesh of IBGP peering sessions.

   ```
   [edit protocols bgp group ibgp]
   user@Border_Router_C# set type internal
   user@Border_Router_C# set local-address 10.255.10.61
   user@Border_Router_C# set neighbor 10.255.10.97
   ```

```
user@Border_Router_C# set neighbor 10.255.10.55
user@Border_Router_C# set neighbor 10.255.10.57
user@Border_Router_C# set neighbor 10.255.10.59
user@Border_Router_C# set export to-bgp
```

4. Configure the multiprotocol BGP-related settings so that the BGP sessions carry the necessary NLRI.

```
[edit protocols bgp group ibgp]
user@Border_Router_C# set family inet unicast
user@Border_Router_C# set family inet-vpn any
user@Border_Router_C# set family inet6 unicast
user@Border_Router_C# set family inet6-vpn any
user@Border_Router_C# set family inet-mvpn signaling
user@Border_Router_C# set family inet6-mvpn signaling
```

5. Configure an interior gateway protocol (IGP).

   This example shows a dual stacking configuration with OSPF and OSPF version 3 configured on the interfaces.

```
[edit protocols ospf3]
user@Border_Router_C# set area 0.0.0.0 interface lo0.0
user@Border_Router_C# set area 0.0.0.0 interface so-1/3/1.0
user@Border_Router_C# set area 0.0.0.0 interface so-0/3/0.0
[edit protocols ospf]
user@Border_Router_C# set traffic-engineering
user@Border_Router_C# set area 0.0.0.0 interface fxp0.0 disable
user@Border_Router_C# set area 0.0.0.0 interface lo0.0
user@Border_Router_C# set area 0.0.0.0 interface so-1/3/1.0
user@Border_Router_C# set area 0.0.0.0 interface so-0/3/0.0
```

6. Configure a global PIM instance on the interface facing the edge device.

   PIM is not configured in the core.

```
[edit protocols pim]
user@Border_Router_C# set rp static address 192.0.2.2
user@Border_Router_C# set rp static address 2::192.0.2.2
user@Border_Router_C# set interface fe-0/1/0.0
user@Border_Router_C# set mpls-internet-multicast
```

7. Configure the ingress replication provider tunnel to create a new unicast tunnel each time a destination needs to be added to the multicast distribution tree.

```
[edit routing-instances test]
user@Border_Router_C# set instance-type mpls-internet-multicast
user@Border_Router_C# set provider-tunnel ingress-replication label-switched-path
user@Border_Router_C# set protocols mvpn
```

NOTE: Alternatively, use the **label-switched-path-template** statement to configure a point-to-point LSP for the ingress tunnel.

Configure the point-to-point LSP to use the default template settings (this is needed only when using RSVP tunnels). For example:

```
[edit routing-instances test provider-tunnel]
user@Border_Router_C# set ingress-replication label-switched-path label-switched-
path-template default-template
user@Border_Router_C# set selective group 203.0.113.0/24 source 192.168.195.145/32
ingress-replication label-switched-path
```

8. Commit the configuration.

```
user@Border_Router_C# commit
```

Results

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

```
user@Border_Router_C# show protocols
mpls {
    ipv6-tunneling;
    interface all;
}
bgp {
    group ibgp {
        type internal;
        local-address 10.255.10.61;
        family inet {
            unicast;
        }
        family inet-vpn {
            any;
        }
        family inet6 {
            unicast;
        }
        family inet6-vpn {
            any;
        }
        family inet-mvpn {
            signaling;
        }
        family inet6-mvpn {
            signaling;
        }
        export to-bgp; ## 'to-bgp' is not defined
        neighbor 10.255.10.97;
        neighbor 10.255.10.55;
        neighbor 10.255.10.57;
        neighbor 10.255.10.59;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface fxp0.0 {
            disable;
        }
        interface lo0.0;
        interface so-1/3/1.0;
        interface so-0/3/0.0;
    }
}
ospf3 {
    area 0.0.0.0 {
        interface lo0.0;
        interface so-1/3/1.0;
        interface so-0/3/0.0;
    }
```

```
}
ldp {
    interface all;
}
pim {
    rp {
        static {
            address 192.0.2.2;
            address 2::192.0.2.2;
        }
    }
    interface fe-0/1/0.0;
    mpls-internet-multicast;
}
```

```

user@Border_Router_C# show routing-instances
test {
    instance-type mpls-internet-multicast;
    provider-tunnel {
        ingress-replication {
            label-switched-path;
        }
    }
    protocols {
        mvpn;
    }
}
```

Verification

IN THIS SECTION

- Checking the Ingress Replication Status on Border Router C | **1090**
- Checking the Routing Table for the MVPN Routing Instance on Border Router C | **1090**
- Checking the MVPN Neighbors on Border Router C | **1092**
- Checking the PIM Join Status on Border Router C | **1093**
- Checking the Multicast Route Status on Border Router C | **1094**
- Checking the Ingress Replication Status on Border Router B | **1094**

- Checking the Routing Table for the MVPN Routing Instance on Border Router B | 1095
- Checking the MVPN Neighbors on Border Router B | 1096
- Checking the PIM Join Status on Border Router B | 1097
- Checking the Multicast Route Status on Border Router B | 1098

Confirm that the configuration is working properly. The following operational output is for LDP ingress replication SPT-only mode. The multicast source behind IP Router B. The multicast receiver is behind IP Router C.

Checking the Ingress Replication Status on Border Router C

Purpose

Use the `show ingress-replication mvpn` command to check the ingress replication status.

Action

```
user@Border_Router_C> show ingress-replication mvpn

Ingress Tunnel: mvpn:1
  Application: MVPN
  Unicast tunnels
    Leaf Address       Tunnel-type      Mode         State
    10.255.10.61       P2P LSP          Existing     Up
```

Meaning

The ingress replication is using a point-to-point LSP, and is in the Up state.

Checking the Routing Table for the MVPN Routing Instance on Border Router C

Purpose

Use the `show route table` command to check the route status.

Action

```
user@Border_Router_C> show route table test.mvpn

test.mvpn.0: 5 destinations, 7 routes (5 active, 1 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:0:0:10.255.10.61/240
                   *[BGP/170] 00:45:55, localpref 100, from 10.255.10.61
                      AS path: I, validation-state: unverified
                    > via so-2/0/1.0
1:0:0:10.255.10.97/240
                   *[MVPN/70] 00:47:19, metric2 1
                      Indirect
5:0:0:32:192.168.195.106:32:198.51.100.1/240
                   *[PIM/105] 00:06:35
                      Multicast (IPv4) Composite
                    [BGP/170] 00:06:35, localpref 100, from 10.255.10.61
                      AS path: I, validation-state: unverified
                    > via so-2/0/1.0
6:0:0:1000:32:192.0.2.2:32:198.51.100.1/240
                   *[PIM/105] 00:07:03
                      Multicast (IPv4) Composite
7:0:0:1000:32:192.168.195.106:32:198.51.100.1/240
                   *[MVPN/70] 00:06:35, metric2 1
                      Multicast (IPv4) Composite
                    [PIM/105] 00:05:35
                      Multicast (IPv4) Composite

test.mvpn-inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:0:0:10.255.10.61/432
                   *[BGP/170] 00:45:55, localpref 100, from 10.255.10.61
                      AS path: I, validation-state: unverified
                    > via so-2/0/1.0
1:0:0:10.255.10.97/432
                   *[MVPN/70] 00:47:19, metric2 1
                      Indirect
```

Meaning

The expected routes are populating the test.mvpn routing table.

Checking the MVPN Neighbors on Border Router C

Purpose

Use the `show mvpn neighbor` command to check the neighbor status.

Action

```
user@Border_Router_C> show mvpn neighbor

MVPN instance:
Legend for provider tunnel
S-    Selective provider tunnel

Legend for c-multicast routes properties (Pr)
DS -- derived from (*, c-g)          RM -- remote VPN route
Family : INET

Instance : test
  MVPN Mode : SPT-ONLY
  Neighbor                                Inclusive Provider Tunnel
  10.255.10.61                            INGRESS-REPLICATION:MPLS Label 16:10.255.10.61

MVPN instance:
Legend for provider tunnel
S-    Selective provider tunnel

Legend for c-multicast routes properties (Pr)
DS -- derived from (*, c-g)          RM -- remote VPN route
Family : INET6

Instance : test
  MVPN Mode : SPT-ONLY
  Neighbor                                Inclusive Provider Tunnel
  10.255.10.61                            INGRESS-REPLICATION:MPLS Label 16:10.255.10.61
```

Checking the PIM Join Status on Border Router C

Purpose

Use the `show pim join extensive` command to check the PIM join status.

Action

```
user@Border_Router_C> show pim join extensive
Instance: PIM.master Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 198.51.100.1
    Source: *
    RP: 192.0.2.2
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 00:07:49
    Downstream neighbors:
        Interface: ge-3/0/6.0
            192.0.2.2 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 00:07:49 Time since last Join: 00:07:49
    Number of downstream interfaces: 1

Group: 198.51.100.1
    Source: 192.168.195.106
    Flags: sparse
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Local RP, Join to Source, No Prune to RP
    Keepalive timeout: 69
    Uptime: 00:06:21
    Number of downstream interfaces: 0

Instance: PIM.master Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

Checking the Multicast Route Status on Border Router C

Purpose

Use the `show multicast route extensive` command to check the multicast route status.

Action

```
user@Border_Router_C> show multicast route extensive
Instance: master Family: INET

Group: 198.51.100.1
    Source: 192.168.195.106/32
    Upstream interface: lsi.0
    Downstream interface list:
        ge-3/0/6.0
    Number of outgoing interfaces: 1
    Session description: NOB Cross media facilities
    Statistics: 18 kBps, 200 pps, 88907 packets
    Next-hop ID: 1048577
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:07:25

Instance: master Family: INET6
```

Checking the Ingress Replication Status on Border Router B

Purpose

Use the `show ingress-replication mvpn` command to check the ingress replication status.

Action

```
user@Border_Router_B> show ingress-replication mvpn
```

```
Ingress Tunnel: mvpn:1
  Application: MVPN
  Unicast tunnels
    Leaf Address       Tunnel-type      Mode         State
    10.255.10.97       P2P LSP          Existing     Up
```

Meaning

The ingress replication is using a point-to-point LSP, and is in the Up state.

Checking the Routing Table for the MVPN Routing Instance on Border Router B

Purpose

Use the `show route table` command to check the route status.

Action

```
user@Border_Router_B> show route table test.mvpn

test.mvpn.0: 5 destinations, 7 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:0:0:10.255.10.61/240
                   *[MVPN/70] 00:49:26, metric2 1
                      Indirect
1:0:0:10.255.10.97/240
                   *[BGP/170] 00:48:22, localpref 100, from 10.255.10.97
                      AS path: I, validation-state: unverified
                    > via so-1/3/1.0
5:0:0:32:192.168.195.106:32:198.51.100.1/240
                   *[PIM/105] 00:09:02
                      Multicast (IPv4) Composite
                    [BGP/170] 00:09:02, localpref 100, from 10.255.10.97
                      AS path: I, validation-state: unverified
                    > via so-1/3/1.0
7:0:0:1000:32:192.168.195.106:32:198.51.100.1/240
                   *[PIM/105] 00:09:02
                      Multicast (IPv4) Composite
                    [BGP/170] 00:09:02, localpref 100, from 10.255.10.97
                      AS path: I, validation-state: unverified
                    > via so-1/3/1.0

test.mvpn-inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:0:0:10.255.10.61/432
                   *[MVPN/70] 00:49:26, metric2 1
                      Indirect
1:0:0:10.255.10.97/432
                   *[BGP/170] 00:48:22, localpref 100, from 10.255.10.97
                      AS path: I, validation-state: unverified
                    > via so-1/3/1.0
```

Meaning

The expected routes are populating the test.mvpn routing table.

Checking the MVPN Neighbors on Border Router B

Purpose

Use the `show mvpn neighbor` command to check the neighbor status.

Action

```
user@Border_Router_B> show mvpn neighbor

MVPN instance:
Legend for provider tunnel
S-    Selective provider tunnel

Legend for c-multicast routes properties (Pr)
DS -- derived from (*, c-g)          RM -- remote VPN route
Family : INET

Instance : test
  MVPN Mode : SPT-ONLY
  Neighbor                              Inclusive Provider Tunnel
  10.255.10.97                          INGRESS-REPLICATION:MPLS Label 16:10.255.10.97

MVPN instance:
```

```
Legend for provider tunnel
S-    Selective provider tunnel

Legend for c-multicast routes properties (Pr)
DS -- derived from (*, c-g)          RM -- remote VPN route
Family : INET6

Instance : test
  MVPN Mode : SPT-ONLY
  Neighbor                              Inclusive Provider Tunnel
  10.255.10.97                          INGRESS-REPLICATION:MPLS Label 16:10.255.10.97
```

Checking the PIM Join Status on Border Router B

Purpose

Use the `show pim join extensive` command to check the PIM join status.

Action

```
user@Border_Router_B> show pim join extensive
Instance: PIM.master Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 198.51.100.1
    Source: 192.168.195.106
    Flags: sparse,spt
    Upstream interface: fe-0/1/0.0
    Upstream neighbor: Direct
    Upstream state: Local Source
    Keepalive timeout: 0
    Uptime: 00:09:39
    Downstream neighbors:
        Interface: Pseudo-MVPN
            Uptime: 00:09:39 Time since last Join: 00:09:39
    Number of downstream interfaces: 1


Instance: PIM.master Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

Checking the Multicast Route Status on Border Router B

Purpose

Use the `show multicast route extensive` command to check the multicast route status.

Action

```
user@Border_Router_B> show multicast route extensive
Instance: master Family: INET

Group: 198.51.100.1
    Source: 192.168.195.106/32
    Upstream interface: fe-0/1/0.0
    Downstream interface list:
        so-1/3/1.0
    Number of outgoing interfaces: 1
    Session description: NOB Cross media facilities
    Statistics: 18 kBps, 200 pps, 116531 packets
    Next-hop ID: 1048580
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:09:43
```

RELATED DOCUMENTATION

Configuring Routing Instances for an MBGP MVPN

mpls-internet-multicast

ingress-replication

create-new-ucast-tunnel

label-switched-path-template (Multicast)

show ingress-replication mvpn

Mapping VPLS Traffic to Specific LSPs

You can map VPLS traffic to specific LSPs by configuring forwarding table policies. This procedure is optional but can be useful. The following example illustrates how you can map lower priority VPLS routing instances to slower LSPs while mapping other higher priority VPLS routing instances to faster LSPs. In this example configuration, **a-to-b1** and **a-to-c1** are high-priority LSPs between the PE routers, while **a-to-b2** and **a-to-c2** are low-priority LSPs between the PE routers.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

To map VPLS traffic, include the `policy-statement vpls-priority` statement:

```
policy-statement vpls-priority {
    term a {
        from {
            rib mpls.0;
            community company-1;
        }
        then {
            install-nexthop lsp [ a-to-b1 a-to-c1 ];
            accept;
        }
    }
    term b {
        from {
            rib mpls.0;
            community company-2;
        }
        then {
            install-nexthop lsp-regex [ "^a-to-b2$" "^a-to-c2$" ];
            accept;
        }
    }
}
community company-1 members target:11111:1;
community company-2 members target:11111:2;
```

You can include the `policy-statement vpls-priority` statement at the following hierarchy levels:

- `[edit policy-options]`

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

Include the `export` statement to apply the `vpls-priority` policy to the forwarding table:

```
export vpls-priority;
```

You can include this statement at the following hierarchy levels:

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

For more information about how to configure routing policies, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

CHAPTER 38

Configuring Inter-AS VPLS and IRB VPLS

IN THIS CHAPTER

- Example: Configuring Inter-AS VPLS with MAC Processing at the ASBR | **1101**
- Configuring VPLS and Integrated Routing and Bridging | **1137**
- Configuring Integrated Routing and Bridging in a VPLS Instance (MX Series Routers Only) | **1141**
- Configuring L2PT for VPLS | **1142**

Example: Configuring Inter-AS VPLS with MAC Processing at the ASBR

IN THIS SECTION

- Requirements | **1101**
- Overview and Topology | **1102**
- Configuration | **1103**

This example describes how to configure inter-AS Virtual Private LAN Service (VPLS) with MAC processing between BGP-signaled VPLS and LDP-signaled VPLS. This feature is described in RFC 4761 as multi-AS VPLS option E or method E.

This example is organized in the following sections:

Requirements

To support inter-AS VPLS between BGP-signaled VPLS and LDP-signaled VPLS, your network must meet the following hardware and software requirements:

- MX Series or M320 routers for the ASBRs.
- Junos OS Release 9.3 or higher.

- Gigabit Ethernet or 10-Gigabit Ethernet interfaces.

Overview and Topology

IN THIS SECTION

● Topology | **1103**

VPLS is a key enabler for delivering multipoint Ethernet service. Major service providers have implemented IP and MPLS backbones and offer VPLS services to large enterprises. Growing demand requires the VPLS network to scale to support many VPLS customers with multiple sites spread across geographically dispersed regions. BGP-signaled VPLS signaling offers scaling advantages over LDP-signaled VPLS. In some environments there is a need for BGP-signaled VPLS to interoperate with existing LDP-signaled VPLS.

This example shows one way to configure BGP-signaled VPLS interworking with an existing LDP-signaled VPLS network.

The advantages of the configuration are:

- You can interconnect customer sites that are spread across different autonomous systems (ASs).
- LDP-signaled VPLS and BGP-signaled VPLS interworking is supported.
- Because the ASBR supports MAC operations, customer sites can be connected directly to the ASBR.
- The inter-AS link is not restricted to Ethernet interfaces.
- Additional configuration for multihoming is relatively straightforward.

Traffic from the interworking virtual private LAN services is switched at the ASBR. The ASBR does all the data plane operations: flooding, MAC learning, aging, and MAC forwarding for each AS to switch traffic among any customer facing interfaces and between the fully meshed pseudowires in the AS. A single pseudowire is created between the ASBRs across the inter-AS link and the ASBRs forward traffic from the pseudowires in each AS to the peer ASBR.

Each ASBR performs VPLS operations within its own AS and performs VPLS operations with the ASBR in the other AS. The ASBR treats the other AS as a BGP-signaled VPLS site. To establish VPLS pseudowires, VPLS NLRI messages are exchanged across the EBGP sessions on the inter-AS links between the ASBRs.

The sample metro network is configured for LDP-signaled VPLS. The core network is configured for BGP-signaled VPLS.

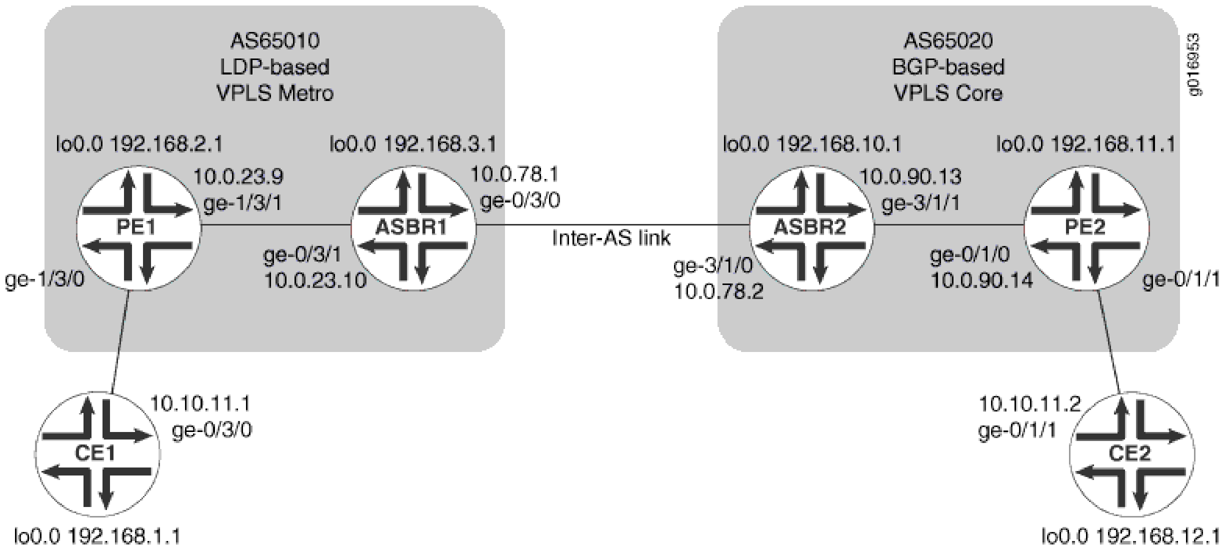
The first part of the example shows the basic configuration steps to configure the logical interfaces, OSPF, internal BGP, LDP, and MPLS. This part of the configuration is the same as other VPLS configurations for LDP-signaled VPLS and BGP-signaled VPLS.

The unique part of the example is configured in the VPLS routing instances, external BGP, and the policy that populates the BGP route table with routes learned from direct routes and OSPF routes. Additional details about the configuration statements are included in the step-by-step procedure.

Figure 83 on page 1103 shows the topology used in this example.

Topology

Figure 83: Inter-AS VPLS with MAC Operations Example Topology



Configuration

IN THIS SECTION

- Configuring Interfaces | 1104
- Configuring OSPF | 1107
- Configuring the Internal BGP Peer Group | 1109
- Configuring LDP | 1111
- Configuring MPLS | 1112
- Configuring the External BGP Peer Group Between the Loopback Interfaces | 1113

- Configuring the External BGP Peer Group Between the Inter-AS Link Interfaces | **1114**
- Configuring the VPLS Routing Instances | **1120**
- Results | **1124**

To configure inter-AS VPLS between BGP-signaled VPLS and LDP-signaled VPLS, perform these tasks.

NOTE: In any configuration session it is a good practice to periodically use the `commit check` command to verify that the configuration can be committed.

Configuring Interfaces

Step-by-Step Procedure

To configure interfaces:

1. On each router, configure an IP address on the loopback logical interface 0 (lo0.0):

```
user@CE1# set interfaces lo0 unit 0 family inet address 192.168.1.1/32 primary
user@PE1# set interfaces lo0 unit 0 family inet address 192.168.2.1/32 primary
user@ASBR1# set interfaces lo0 unit 0 family inet address 192.168.3.1/32 primary
user@ASBR2# set interfaces lo0 unit 0 family inet address 192.168.10.1/32 primary
user@PE2# set interfaces lo0 unit 0 family inet address 192.168.11.1/32 primary
user@CE2# set interfaces lo0 unit 0 family inet address 192.168.12.1/32 primary
```

2. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

3. On each router, display the interface information for `lo0` and verify that the correct IP address is configured:

```
user@host> show interfaces lo0

Physical interface: lo0, Enabled, Physical link is Up
  Interface index: 6, SNMP ifIndex: 6
  Type: Loopback, MTU: Unlimited
  Device flags   : Present Running Loopback
  Interface flags: SNMP-Traps
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface lo0.0 (Index 75) (SNMP ifIndex 16)
    Flags: SNMP-Traps Encapsulation: Unspecified
    Input packets : 0
    Output packets: 0
    Protocol inet, MTU: Unlimited
      Flags: None
      Addresses
        Local: 127.0.0.1
      Addresses, Flags: Primary Is-Default Is-Primary
        Local: 192.168.3.1
Logical interface lo0.16384 (Index 64) (SNMP ifIndex 21)
    Flags: SNMP-Traps Encapsulation: Unspecified
```

```
      Input packets : 0
      Output packets: 0
      Protocol inet, MTU: Unlimited
        Flags: None
        Addresses
          Local: 127.0.0.1

    Logical interface lo0.16385 (Index 65) (SNMP ifIndex 22)
      Flags: SNMP-Traps Encapsulation: Unspecified
      Input packets : 0
      Output packets: 0
      Protocol inet, MTU: Unlimited
        Flags: None
```

In the example above notice that the primary `lo0` local address for the `inet` protocol family on Router ASBR1 is `192:168:3:1`.

4. On each router, configure an IP address and protocol family on the Gigabit Ethernet interfaces. Specify the `inet` protocol family.

```
user@CE1# set interfaces ge-0/3/0 unit 0 family inet address 10.10.11.1/24
user@PE1# set interfaces ge-1/3/1 unit 0 family inet address 10.0.23.9/30
user@ASBR1# set interfaces ge-0/3/1 unit 0 family inet address 10.0.23.10/30
user@ASBR1# set interfaces ge-0/3/0 unit 0 family inet address 10.0.78.1/30
user@ASBR2# set interfaces ge-3/1/0 unit 0 family inet address 10.0.78.2/30
user@ASBR2# set interfaces ge-3/1/1 unit 0 family inet address 10.0.90.13/30
user@PE2# set interfaces ge-0/1/0 unit 0 family inet address 10.0.90.14/30
user@CE2# set interfaces ge-0/1/1 unit 0 family inet address 10.10.11.2/24
```

5. On each router, commit the configuration:

```
user@host> **commit check**
```

```
configuration check succeeds
```

```
user@host> **commit**
```

```
commit complete
```

6. Display information for Gigabit Ethernet interfaces and verify that the IP address and protocol family are configured correctly.

```
user@ASBR2> show interfaces ge-* terse
Interface               Admin Link Proto    Local                 Remote
ge-3/1/0                up    up
ge-3/1/0.0              up    up   inet     10.0.78.2/30
                               multiservice
ge-3/1/1                up    up
ge-3/1/1.0              up    up   inet     10.0.90.13/30
                               multiservice
ge-3/1/2                up    down
ge-3/1/3                up    down
```

Configuring OSPF

Step-by-Step Procedure

To configure OSPF:

1. On the PE and ASBR routers, configure the provider instance of OSPF. Configure OSPF traffic engineering support. Specify area 0.0.0.1 in the LDP-signaled VPLS network and area 0.0.0.0 in the

BGP-signaled network. Specify the Gigabit Ethernet logical interfaces between the PE and ASBR routers. Specify `lo0.0` as a passive interface.

```
user@PE1# set protocols ospf traffic-engineering
user@PE1# set protocols ospf area 0.0.0.1 interface ge-1/3/1.0
user@PE1# set protocols ospf area 0.0.0.1 interface lo0.0 passive
user@ASBR1# set protocols ospf traffic-engineering
user@ASBR1# set protocols ospf area 0.0.0.1 interface ge-0/3/1.0
user@ASBR1# set protocols ospf area 0.0.0.1 interface lo0.0 passive
user@ASBR2# set protocols ospf traffic-engineering
user@ASBR2# set protocols ospf area 0.0.0.0 interface ge-3/1/1.0
user@ASBR2# set protocols ospf area 0.0.0.0 interface lo0.0 passive
user@PE2# set protocols ospf traffic-engineering
user@PE2# set protocols ospf area 0.0.0.0 interface ge-0/1/0.0
user@PE2# set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

2. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

3. Display OSPF neighbor information and verify that the PE routers form adjacencies with the ASBR router in the same area. Verify that the neighbor state is `Full`.

```
user@host> show ospf neighbor
Address          Interface          State     ID              Pri  Dead
10.0.23.10       ge-1/3/1.0         Full      192.168.3.1     128  31
```

Configuring the Internal BGP Peer Group

Step-by-Step Procedure

The purpose of configuring an internal BGP peer group is to create a full mesh of BGP LSPs among the PE routers in the BGP-signaled AS, including the ASBR routers.

To configure the internal BGP peer group:

1. The purpose of this step is to create a full mesh of IBGP peers between the PE routers, including the ASBR routers, within the BGP-signaled AS.

   On Router ASBR2, configure internal BGP. Specify the BGP type as `internal`. Specify the local address as the local `lo0` IP address.

   Specify the `inet` protocol family. Specify the `labeled-unicast` statement and the `resolve-vpn` option. The `labeled-unicast` statement causes the router to advertise labeled routes out of the IPv4 inet.0 route table and places labeled routes into the inet.0 route table. The `resolve-vpn` option puts labeled routes in the MPLS inet.3 route table The inet.3 route table is used to resolve routes for the PE router located in the other AS.

   Specify the `l2vpn` family to indicate to the router that this is a VPLS. Specify the `signaling` option to configure BGP as the signaling protocol. This enables BGP to carry Layer 2 VPLS NLRI messages for this peer group.

   Specify the `lo0` interface IP address of the PE as the neighbor. Configure an autonomous system identifier.

   ```
   user@ASBR2# set protocols bgp group core-ibgp type internal
   user@ASBR2# set protocols bgp group core-ibgp local-address 192.168.10.1
   user@ASBR2# set protocols bgp group core-ibgp family inet labeled-unicast resolve-vpn
   user@ASBR2# set protocols bgp group core-ibgp family l2vpn signaling
   user@ASBR2# set protocols bgp group core-ibgp neighbor 192.168.11.1
   user@ASBR2# set routing-options autonomous-system 0.65020
   ```

2. On Router PE2, configure internal BGP. Specify the BGP type as `internal`. Specify the local address as the local `lo0` IP address.

   Specify the `l2vpn` family to indicate this is a VPLS. Specify the `signaling` option to configure BGP as the signaling protocol. This enables BGP to carry Layer 2 VPLS NLRI messages.

Specify the `lo0` interface IP address of Router ASBR2 as the neighbor. Configure an autonomous system identifier.

```
user@PE2# set protocols bgp group core-ibgp type internal
user@PE2# set protocols bgp group core-ibgp local-address 192.168.11.1
user@PE2# set protocols bgp group core-ibgp family l2vpn signaling
user@PE2# set protocols bgp group core-ibgp neighbor 192.168.10.1
user@PE2# set routing-options autonomous-system 0.65020
```

3. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

4. On Router PE2 and Router ASBR2, display BGP neighbor information and verify that the peer connection state is `Established`.

```
user@ASBR2> show bgp neighbor
Peer: 192.168.11.1+49443 AS 65020 Local: 192.168.10.1+179 AS 65020
  Type: Internal    State: Established    Flags: ImportEval Sync
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Options: Preference LocalAddress AddressFamily Rib-group Refresh
  Address families configured: l2vpn-signaling inet-labeled-unicast
  Local Address: 192.168.10.1 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 192.168.11.1     Local ID: 192.168.10.1     Active Holdtime: 90
  Keepalive Interval: 30         Peer index: 0

...
```

Configuring LDP

Step-by-Step Procedure

To configure LDP:

1. On the PE and ASBR routers, configure LDP with the Gigabit Ethernet interfaces between the PE and ASBR routers, and between the two ASBR routers. To support LDP-signaled VPLS, additionally configure LDP with the `lo0.0` interface on Router PE1 and Router ASBR1:

```
user@PE1# set protocols ldp interface ge-1/3/1.0
user@PE1# set protocols ldp interface lo0.0
user@ASBR1# set protocols ldp interface ge-0/3/1.0
user@ASBR1# set protocols ldp interface ge-0/3/0.0
user@ASBR1# set protocols ldp interface lo0.0
user@ASBR2# set protocols ldp interface ge-3/1/0.0
user@ASBR2# set protocols ldp interface ge-3/1/1.0
user@PE2# set protocols ldp interface ge-0/1/0.0
```

NOTE: The configuration of LDP signaling between the ASBR routers is not required for Inter-AS VPLS. It is included here for reference only and might be used in LDP environments.

2. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

3. Display LDP configuration information and verify that the correct interfaces are configured. LDP operation can be verified after MPLS is configured.

```
user@ASBR1> show configuration protocols ldp
interface ge-0/3/0.0;
interface ge-0/3/1.0;
interface lo0.0;
```

   The preceding example is from ASBR1.

Configuring MPLS

Step-by-Step Procedure

To configure MPLS:

1. On the PE and ASBR routers, configure MPLS. Enable MPLS on the logical interfaces. Add the Gigabit Ethernet interfaces to the MPLS protocol. This adds entries to the MPLS forwarding table.

```
user@PE1# set protocols mpls interface ge-1/3/1.0
user@PE1# set interfaces ge-1/3/1 unit 0 family mpls
user@ASBR1# set protocols mpls interface ge-0/3/1.0
user@ASBR1# set protocols mpls interface ge-0/3/0.0
user@ASBR1# set interfaces ge-0/3/1 unit 0 family mpls
user@ASBR1# set interfaces ge-0/3/0 unit 0 family mpls
user@ASBR2# set protocols mpls interface ge-3/1/0.0
user@ASBR2# set protocols mpls interface ge-3/1/1.0
user@ASBR2# set interfaces ge-3/1/0 unit 0 family mpls
user@ASBR2# set interfaces ge-3/1/1 unit 0 family mpls
user@PE2# set protocols mpls interface ge-0/1/0.0
user@PE2# set interfaces ge-0/1/0 unit 0 family mpls
```

2. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

3. On the PE and ASBR routers, display LDP neighbor information and verify that the directly connected LDP neighbors are listed:

```
user@ASBR1> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.168.2.1        lo0.0              192.168.2.1:0            44
10.0.78.2          ge-0/3/0.0         192.168.10.1:0           13
10.0.23.9          ge-0/3/1.0         192.168.2.1:0            11
```

   The preceding example is from ASBR1.

Configuring the External BGP Peer Group Between the Loopback Interfaces

Step-by-Step Procedure

To configure the external BGP (EBGP) peer group between the loopback interfaces:

1. On Router ASBR1 and Router PE1, configure an autonomous system identifier:

```
user@PE1# set routing-options autonomous-system 0.65010
user@ASBR1# set routing-options autonomous-system 0.65010
```

2. On Router ASBR1, configure an external BGP peer group for the loopback interfaces. Specify the `external` BGP group type. Include the `multihop` statement. Specify the local address as the local `lo0` IP

address. Configure the `l2vpn` family for BGP signaling. Configure the peer AS as the core AS number. Specify the `lo0` IP address of Router ASBR2 as the neighbor.

```
user@ASBR1# set protocols bgp group vpls-core type external
user@ASBR1# set protocols bgp group vpls-core multihop
user@ASBR1# set protocols bgp group vpls-core local-address 192.168.3.1
user@ASBR1# set protocols bgp group vpls-core family l2vpn signaling
user@ASBR1# set protocols bgp group vpls-core peer-as 65020
user@ASBR1# set protocols bgp group vpls-core neighbor 192.168.10.1
```

3. On Router ASBR2, configure an external BGP peer group for the loopback interfaces. Specify the `external` BGP group type. Include the `multihop` statement. The `multihop` statement is needed because the EBGP neighbors are in different ASs. Specify the local address as the local `lo0` IP address. Configure the `l2vpn` family for BGP signaling. Configure the peer AS as the metro AS number. Specify the `lo0` IP address of Router ASBR1 as the neighbor.

```
user@ASBR2# set protocols bgp group vpls-metro type external
user@ASBR2# set protocols bgp group vpls-metro multihop
user@ASBR2# set protocols bgp group vpls-metro local-address 192.168.10.1
user@ASBR2# set protocols bgp group vpls-metro family l2vpn signaling
user@ASBR2# set protocols bgp group vpls-metro peer-as 65010
user@ASBR2# set protocols bgp group vpls-metro neighbor 192.168.3.1
```

4. On each router, commit the configuration:

```
user@host> commit
```

Configuring the External BGP Peer Group Between the Inter-AS Link Interfaces

Step-by-Step Procedure

The purpose of configuring external BGP peer groups between the inter-AS link interfaces is to create a full mesh of BGP LSPs among the ASBR routers. To configure the external BGP peer group between the inter-AS link interfaces:

1. On Router ASBR1, configure a policy to export OSPF and direct routes, including the `lo0` address of the PE routers, into BGP for the establishment of label-switched paths (LSPs):

   ```
   user@ASBR1# set policy-options policy-statement loopback term term1 from protocol ospf
   user@ASBR1# set policy-options policy-statement loopback term term1 from protocol direct
   user@ASBR1# set policy-options policy-statement loopback term term1 from route-filter
   192.168.0.0/16 longer
   user@ASBR1# set policy-options policy-statement loopback term term1 then accept
   ```

2. On Router ASBR1, configure an external BGP peer group for the inter-AS link. Specify the `external` BGP group type. Specify the local inter-AS link IP address as the local address. Configure the `inet` family and include the `labeled-unicast` and `resolve-vpn` statements. The `labeled-unicast` statement advertises labeled routes out of the IPv4 inet.0 route table and places labeled routes into the inet.0 route table. The `resolve-vpn` option stores labeled routes in the MPLS `inet.3` route table.

   Include the `export` statement and specify the policy you created. Configure the peer AS as the core AS number. Specify the inter-AS link IP address of Router ASBR2 as the neighbor.

   ```
   user@ASBR1# set protocols bgp group metro-core type external
   user@ASBR1# set protocols bgp group metro-core local-address 10.0.78.1
   user@ASBR1# set protocols bgp group metro-core family inet labeled-unicast resolve-vpn
   user@ASBR1# set protocols bgp group metro-core export loopback
   user@ASBR1# set protocols bgp group metro-core peer-as 65020
   user@ASBR1# set protocols bgp group metro-core neighbor 10.0.78.2
   ```

3. On Router ASBR2, configure a policy to export OSPF and direct routes, including the `lo0` address, into BGP for the establishment of LSPs:

   ```
   user@ASBR2# set policy-options policy-statement loopback term term1 from protocol ospf
   user@ASBR2# set policy-options policy-statement loopback term term1 from protocol direct
   user@ASBR2# set policy-options policy-statement loopback term term1 from route-filter
   192.168.0.0/16 longer
   user@ASBR2# set policy-options policy-statement loopback term term1 then accept
   ```

4. On Router ASBR2, configure an external BGP peer group for the inter-AS link. Specify the `external` BGP group type. Specify the local inter-AS link IP address as the local address. Configure the `inet` family and include the `labeled-unicast` and `resolve-vpn` statements. Include the `export` statement and

specify the policy you created. Configure the peer AS as the core AS number. Specify the inter-AS link IP address of Router ASBR1 as the neighbor.

```
user@ASBR2# set protocols bgp group core-metro type external
user@ASBR2# set protocols bgp group core-metro local-address 10.0.78.2
user@ASBR2# set protocols bgp group core-metro family inet labeled-unicast resolve-vpn
user@ASBR2# set protocols bgp group core-metro export loopback
user@ASBR2# set protocols bgp group core-metro peer-as 65010
user@ASBR2# set protocols bgp group core-metro neighbor 10.0.78.1
```

5. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

6. On Router ASBR1, display the BGP neighbors. Verify that the first peer is the IP address of the Gigabit Ethernet interface of Router ASBR2. Verify that the second peer is the IP address of the `lo0` interface of Router ASBR2. Also verify that the state of each peer is `Established`. Notice that on Router ASBR1 the NLRI advertised by Router ASBR2 the inter-AS link peer is `inet-labeled-unicast` and the NLRI advertised by Router ASBR2 the loopback interface peer is `l2vpn-signaling`.

```
user@ASBR1> show bgp neighbor
Peer: 10.0.78.2+65473 AS 65020 Local: 10.0.78.1+179 AS 65010
  Type: External    State: Established    Flags: Sync
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: Cease
  Export: [ loopback ]
  Options: Preference LocalAddress AddressFamily PeerAS Rib-group Refresh
  Address families configured: inet-labeled-unicast
  Local Address: 10.0.78.1 Holdtime: 90 Preference: 170
  Number of flaps: 3
  Last flap event: Stop
  Error: 'Cease' Sent: 1 Recv: 2
  Peer ID: 192.168.10.1     Local ID: 192.168.3.1      Active Holdtime: 90
  Keepalive Interval: 30         Peer index: 0
  BFD: disabled, down
  Local Interface: ge-0/3/0.0
  NLRI for restart configured on peer: inet-labeled-unicast
  NLRI advertised by peer: inet-labeled-unicast
  NLRI for this session: inet-labeled-unicast
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Restart time requested by this peer: 120
  NLRI that peer supports restart for: inet-labeled-unicast
  NLRI that restart is negotiated for: inet-labeled-unicast
  NLRI of received end-of-rib markers: inet-labeled-unicast
  NLRI of all end-of-rib markers sent: inet-labeled-unicast
  Peer supports 4 byte AS extension (peer-as 65020)
  Table inet.0 Bit: 10000
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              2
    Received prefixes:            3
    Accepted prefixes:            3
    Suppressed due to damping:    0
    Advertised prefixes:          3
  Last traffic (seconds): Received 8    Sent 3    Checked 60
  Input messages:  Total 8713   Updates 3       Refreshes 0     Octets 165688
  Output messages: Total 8745   Updates 2       Refreshes 0     Octets 166315
  Output Queue[0]: 0

Peer: 192.168.10.1+51234 AS 65020 Local: 192.168.3.1+179 AS 65010
  Type: External    State: Established    Flags: Sync
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: Cease
  Options: Multihop Preference LocalAddress AddressFamily PeerAS Rib-group Refresh
  Address families configured: l2vpn-signaling
  Local Address: 192.168.3.1 Holdtime: 90 Preference: 170
  Number of flaps: 3
  Last flap event: Stop
  Error: 'Cease' Sent: 1 Recv: 2
  Peer ID: 192.168.10.1     Local ID: 192.168.3.1      Active Holdtime: 90
  Keepalive Interval: 30         Peer index: 0
  BFD: disabled, down
  NLRI for restart configured on peer: l2vpn-signaling
  NLRI advertised by peer: l2vpn-signaling
  NLRI for this session: l2vpn-signaling
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Restart time requested by this peer: 120
  NLRI that peer supports restart for: l2vpn-signaling
  NLRI that restart is negotiated for: l2vpn-signaling
  NLRI of received end-of-rib markers: l2vpn-signaling
  NLRI of all end-of-rib markers sent: l2vpn-signaling
  Peer supports 4 byte AS extension (peer-as 65020)
  Table bgp.l2vpn.0 Bit: 20000
    RIB State: BGP restart is complete
    RIB State: VPN restart is complete
    Send state: in sync
    Active prefixes:              1
    Received prefixes:            1
    Accepted prefixes:            1
    Suppressed due to damping:    0
    Advertised prefixes:          1
  Table inter-as.l2vpn.0
    RIB State: BGP restart is complete
    RIB State: VPN restart is complete
    Send state: not advertising
    Active prefixes:              1
    Received prefixes:            1
    Accepted prefixes:            1
    Suppressed due to damping:    0
  Last traffic (seconds): Received 19   Sent 18   Checked 42
  Input messages:  Total 8712   Updates 3       Refreshes 0     Octets 165715
  Output messages: Total 8744   Updates 2       Refreshes 0     Octets 166342
  Output Queue[1]: 0
  Output Queue[2]: 0
```

7. On Router ASBR2, display the BGP summary. Notice that the first peer is the IP address of the Gigabit Ethernet interface of Router ASBR1, the second peer is the IP address of the `lo0` interface of

Router ASBR1, and the third peer is the `lo0` interface of Router PE2. Verify that the state of each peer is `Established`.

```
user@ASBR2> show bgp summary
Groups: 3 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0                 3          2          0          0          0          0
bgp.l2vpn.0            2          2          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.0.78.1             65010       8781       8748       0       2 2d 17:54:56 Establ
  inet.0: 2/3/3/0
192.168.3.1           65010       8780       8747       0       2 2d 17:54:54 Establ
  bgp.l2vpn.0: 1/1/1/0
  inter-as.l2vpn.0: 1/1/1/0
192.168.11.1          65020       8809       8763       0       1 2d 17:59:22 Establ
  bgp.l2vpn.0: 1/1/1/0
  inter-as.l2vpn.0: 1/1/1/0
```

8. On Router PE2, display the BGP group. Verify that the peer is the IP address of the `lo0` interface of Router ASBR2. Verify that the number of established peer sessions is 1.

```
user@PE1> show bgp group
Group Type: Internal    AS: 65020                  Local AS: 65020
  Name: core-ibgp       Index: 1                   Flags: Export Eval
  Holdtime: 0
  Total peers: 1        Established: 1
  192.168.10.1+179
  bgp.l2vpn.0: 1/1/1/0
  inter-as.l2vpn.0: 1/1/1/0

Groups: 1  Peers: 1    External: 0    Internal: 1    Down peers: 0   Flaps: 7
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0            1          1          0          0          0          0
inte.l2vpn.0           1          1          0          0          0          0
```

Configuring the VPLS Routing Instances

Step-by-Step Procedure

To configure the VPLS routing instances:

1. On Router PE1, configure the VPLS routing instance. To enable a VPLS instance, specify the `vpls` instance type. Configure VPLS on the CE-facing Gigabit Ethernet interface. Configure the CE-facing interface to use `ethernet-vpls` encapsulation.

   ```
   user@PE1# set routing-instances metro instance-type vpls
   user@PE1# set routing-instances metro interface ge-1/3/0.0
   ```

2. On Router PE1, configure the VPLS protocol within the routing instance. To uniquely identify the virtual circuit, configure the VPLS identifier. The VPLS identifier uniquely identifies each VPLS in the router. Configure the same VPLS ID on all the routers for a given VPLS.

   Specify the IP address of the `lo0` interface on Router ASBR2 as the neighbor.

   Configure the CE-facing interface to use `ethernet-vpls` encapsulation and the `vpls` protocol family.

   ```
   user@PE1# set routing-instances metro protocols vpls vpls-id 101
   user@PE1# set routing-instances metro protocols vpls neighbor 192.168.3.1
   user@PE1# set interfaces ge-1/3/0 encapsulation ethernet-vpls
   user@PE1# set interfaces ge-1/3/0 unit 0 family vpls
   ```

3. On Router ASBR1, configure the VPLS routing instance. To enable a VPLS instance, specify the `vpls` instance type. Configure a route distinguisher and a VRF target. The `vrf-target` statement causes default VRF import and export policies to be generated that accept and tag routes with the specified target community.

   NOTE: A route distinguisher allows the router to distinguish between two identical IP prefixes used as VPN routes. Configure a different route distinguisher on each ASBR router.

NOTE: You must configure the same VRF target on both ASBR routers.

```
user@ASBR1# set routing-instances inter-as instance-type vpls
user@ASBR1# set routing-instances inter-as route-distinguisher 65010:1
user@ASBR1# set routing-instances inter-as vrf-target target:2:1
```

4. On Router ASBR1, configure the VPLS protocol within the routing instance.

   Configure the VPLS identifier. Specify the IP address of the `lo0` interface on Router PE1 as the neighbor.

```
user@ASBR1# set routing-instances inter-as protocols vpls vpls-id 101
user@ASBR1# set routing-instances inter-as protocols vpls neighbor 192.168.2.1
```

NOTE: The VPLS identifier uniquely identifies each LDP-signaled VPLS in the router. Configure the same VPLS ID on Router PE1 and Router ASBR1.

5. On Router ASBR1, configure the VPLS site within the routing instance. Configure the site identifier as required by the protocol to establish the EBGP pseudowire. As a best practice for more complex topologies involving multihoming, configure a site preference.

```
user@ASBR1# set routing-instances inter-as protocols vpls site ASBR-metro site-identifier 1
user@ASBR1# set routing-instances inter-as protocols vpls site ASBR-metro site-preference
10000
```

6. On Router ASBR1, configure the VPLS mesh group `peer-as` statement within the routing instance to specify which ASs belong to this AS mesh group. Configure the peer AS for the mesh group as `all`.

   This statement enables the router to establish a single pseudowire between the ASBR routers. VPLS NLRI messages are exchanged across the EBGP sessions on the inter-AS links between the ASBR routers. All autonomous systems are in one mesh group.

```
user@ASBR1# set routing-instances inter-as protocols vpls mesh-group metro peer-as all
```

7. On ASBR2, configure the VPLS routing instance. To enable a VPLS instance, specify the `vpls` instance type. Configure a route distinguisher and a VRF target. The `vrf-target` statement causes

default VRF import and export policies to be generated that accept and tag routes with the specified target community.

NOTE: A route distinguisher allows the router to distinguish between two identical IP prefixes used as VPN routes. Configure a different route distinguisher on each ASBR router.

NOTE: You must configure the same VRF target community on both ASBR routers.

```
user@ASBR2# set routing-instances inter-as instance-type vpls
user@ASBR2# set routing-instances inter-as route-distinguisher 65020:1
user@ASBR2# set routing-instances inter-as vrf-target target:2:1
```

8. On Router ASBR2, configure the VPLS site within the routing instance. Configure the site identifier as required by the protocol.

```
user@ASBR2# set routing-instances inter-as protocols vpls site ASBR-core site-identifier 2
```

9. On Router ASBR2, configure the VPLS mesh group within the routing instance to specify which VPLS PEs belong to this AS mesh group. Configure the peer AS for the mesh group as `all`.

   This statement enables the router to establish a single pseudowire between the ASBR routers. VPLS NLRI messages are exchanged across the EBGP sessions on the inter-AS links between the ASBR routers. All autonomous systems are in one mesh group.

```
user@ASBR1# set routing-instances inter-as protocols vpls mesh-group core peer-as all
```

10. On Router PE2, configure the VPLS routing instance. To enable a VPLS instance, specify the `vpls` instance type. Configure VPLS on the CE-facing Gigabit Ethernet interface. Configure a route distinguisher and a VRF target.

```
user@PE2# set routing-instances inter-as instance-type vpls
user@PE2# set routing-instances inter-as interface ge-0/1/1.0
user@PE2# set routing-instances inter-as route-distinguisher 65020:1
user@PE2# set routing-instances inter-as vrf-target target:2:1
```

11. On Router PE2, configure the VPLS site within the routing instance. Configure the site identifier as required by the protocol.

    Configure the CE-facing interface to use `ethernet-vpls` encapsulation and the `vpls` protocol family.

```
user@PE2# set routing-instances inter-as protocols vpls site PE2 site-identifier 3
user@PE2# set interfaces ge-0/1/1 encapsulation ethernet-vpls
user@PE2# set interfaces ge-0/1/1 unit 0 family vpls
```

12. On each router, commit the configuration:

```
user@host> commit check
```

```
configuration check succeeds
```

```
user@host> commit
```

```
commit complete
```

13. On the PE routers, display the CE-facing Gigabit Ethernet interface information and verify that the encapsulation is configured correctly:

```
user@host> show interfaces ge-1/3/0
Address            Interface          Label space ID         Hold time
10.0.23.10         ge-1/3/1.0         192.168.3.1:0            11

Physical interface: ge-1/3/0, Enabled, Physical link is Up
  Interface index: 147, SNMP ifIndex: 145
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, MAC-REWRITE Error: None,
  Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 4 supported, 4 maximum usable queues
  Schedulers     : 256
  Current address: 00:12:1e:ee:34:db, Hardware address: 00:12:1e:ee:34:db
  Last flapped   : 2008-08-27 19:02:52 PDT (5d 22:32 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Ingress rate at Packet Forwarding Engine      : 0 bps (0 pps)
  Ingress drop rate at Packet Forwarding Engine : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None

  Logical interface ge-1/3/0.0 (Index 84) (SNMP ifIndex 146)
    Flags: SNMP-Traps Encapsulation: ENET2
    Input packets : 0
    Output packets: 1
    Protocol inet, MTU: 1500
      Flags: None
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.10.11/24, Local: 10.10.11.11, Broadcast: 10.10.11.255
```

Results

This section describes commands you can use to test the operation of the VPLS.

1. To verify the VPLS connections have been established, enter the `show vpls connections` command on Router PE 1.

```
user@PE1> show vpls connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
```

```
BK -- Backup connection          ST -- Standby connection

Legend for interface status
Up -- operational
Dn -- down

Instance: metro
  VPLS-id: 101
    Neighbor                  Type  St     Time last up          # Up trans
    192.168.3.1(vpls-id 101)  rmt   Up     Sep  9 14:05:18 2008           1
      Remote PE: 192.168.3.1, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 800000
      Local interface: vt-1/2/0.1048576, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls metro neighbor 192.168.3.1 vpls-id 101
```

In the display from Router PE1, verify that the neighbor is the `lo0` address of Router ASBR1 and that the status is `Up`.

2. To verify the VPLS connections have been established, enter the `show vpls connections` command on Router ASBR 1.

```
user@ASBR1> show vpls connections
...
Instance: inter-as
  BGP-VPLS State
  Mesh-group connections: metro
    Neighbor         Local-site   Remote-site   St      Time last up
    192.168.10.1     1            2             Up      Sep  8 20:16:28 2008
      Incoming label: 800257, Outgoing label: 800000
      Local interface: vt-1/2/0.1049088, Status: Up, Encapsulation: VPLS
  LDP-VPLS State
  VPLS-id: 101
  Mesh-group connections: __ves__
    Neighbor                  Type  St     Time last up          # Up trans
    192.168.2.1(vpls-id 101)  rmt   Up     Sep  9 14:05:22 2008           1
      Remote PE: 192.168.2.1, Negotiated control-word: No
      Incoming label: 800000, Outgoing label: 800001
      Local interface: vt-0/1/0.1049089, Status: Up, Encapsulation: ETHERNET
        Description: Intf - vpls inter-as neighbor 192.168.2.1 vpls-id 101
```

In the display from Router ASBR1, verify that the neighbor is the `lo0` address of Router PE1 and that the status is `Up`.

3. To verify the VPLS connections have been established, enter the `show vpls connections` command on Router ASBR2.

```
user@ASBR2> show vpls connections
...
Instance: inter-as
  BGP-VPLS State
  Mesh-group connections: __ves__
    Neighbor         Local-site   Remote-site   St      Time last up
    192.168.11.1     2            3             Up      Sep 11 15:18:23 2008
      Incoming label: 800002, Outgoing label: 800001
      Local interface: vt-4/0/0.1048839, Status: Up, Encapsulation: VPLS
  Mesh-group connections: core
    Neighbor         Local-site   Remote-site   St      Time last up
    192.168.3.1      2            1             Up      Sep  8 20:16:28 2008
      Incoming label: 800000, Outgoing label: 800257
      Local interface: vt-4/0/0.1048834, Status: Up, Encapsulation: VPLS
```

In the display from Router ASBR2, verify that the neighbor is the `lo0` address of Router PE2 and that the status is `Up`.

4. To verify the VPLS connections have been established, enter the `show vpls connections` command on Router PE2.

```
user@PE2> show vpls connections
...
Instance: inter-as
  Local site: PE2 (3)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Sep  8 20:16:28 2008           1
      Remote PE: 192.168.10.1, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: 800002
      Local interface: vt-0/3/0.1048832, Status: Up, Encapsulation: VPLS
        Description: Intf - vpls inter-as local site 3 remote site 2
```

In the display from Router PE2, verify that the remote PE is the `lo0` address of Router ASBR2 and that the status is `Up`.

5. To verify that the CE routers can send and receive traffic across the VPLS, use the `ping` command.

```
user@CE1> ping 10.10.11.2
PING 10.10.11.2 (10.10.11.2): 56 data bytes
64 bytes from 10.10.11.2: icmp_seq=0 ttl=64 time=1.369 ms
64 bytes from 10.10.11.2: icmp_seq=1 ttl=64 time=1.360 ms
64 bytes from 10.10.11.2: icmp_seq=2 ttl=64 time=1.333 ms
^C
```

```
user@CE2> ping 10.10.11.1
PING 10.10.11.1 (10.10.11.1): 56 data bytes
64 bytes from 10.10.11.1: icmp_seq=0 ttl=64 time=6.209 ms
64 bytes from 10.10.11.1: icmp_seq=1 ttl=64 time=1.347 ms
64 bytes from 10.10.11.1: icmp_seq=2 ttl=64 time=1.324 ms
^C
```

If Router CE1 can send traffic to and receive traffic from Router CE2 and Router CE2 can send traffic to and receive traffic from Router CE1, the VPLS is performing correctly.

6. To display the configuration for Router CE1, use the `show configuration` command.

For your reference, the relevant sample configuration for Router CE1 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.1.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.11.1/24;
            }
        }
    }
}
```

7. To display the configuration for Router PE1, use the `show configuration` command.

   For your reference, the relevant sample configuration for Router PE1 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.2.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-1/3/0 {
        encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    }
    ge-1/3/1 {
        unit 0 {
            family inet {
                address 10.0.23.9/30;
            }
            family mpls;
        }
    }
}
routing-options {
    autonomous-system 0.65010;
}
protocols {
    mpls {
        interface ge-1/3/1.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.1 {
            interface ge-1/3/1.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface ge-1/3/1.0;
        interface lo0.0;
    }
}
routing-instances {
    metro {
        instance-type vpls;
        interface ge-1/3/0.0;
        protocols {
            vpls {
                vpls-id 101;
                neighbor 192.168.3.1;
            }
        }
    }
}
```

8. To display the configuration for Router ASBR1, use the `show configuration` command.

   For your reference, the relevant sample configuration for Router ASBR1 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.3.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-0/3/0 {
        unit 0 {
            family inet {
                address 10.0.78.1/30;
            }
            family mpls;
        }
    }
    ge-0/3/1 {
        unit 0 {
            family inet {
                address 10.0.23.10/30;
            }
            family mpls;
        }
    }
}
routing-options {
    autonomous-system 0.65010;
}
protocols {
    mpls {
        interface ge-0/3/1.0;
        interface ge-0/3/0.0;
    }
    bgp {
        group vpls-core {
            type external;
            multihop;
            local-address 192.168.3.1;
            family l2vpn {
                signaling;
            }
            peer-as 65020;
            neighbor 192.168.10.1;
        }
        group metro-core {
            type external;
            local-address 10.0.78.1;
            family inet {
                labeled-unicast {
                    resolve-vpn;
                }
            }
            export loopback;
            peer-as 65020;
            neighbor 10.0.78.2;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.1 {
            interface ge-0/3/1.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface ge-0/3/0.0;
        interface ge-0/3/1.0;
        interface lo0.0;
    }
}
policy-options {
    policy-statement loopback {
        term term1 {
            from {
                protocol [ ospf direct ];
                route-filter 192.168.0.0/16 longer;
            }
            then accept;
        }
    }
}
routing-instances {
    inter-as {
        instance-type vpls;
        route-distinguisher 65010:1;
        vrf-target target:2:1;
        protocols {
            vpls {
                site ASBR-metro {
                    site-identifier 1;
                    site-preference 10000;
                }
                vpls-id 101;
                neighbor 192.168.2.1;
                mesh-group metro {
                    peer-as {
                        all;
                    }
                }
            }
        }
    }
}
```

9. To display the configuration for Router ASBR2, use the `show configuration` command.

   For your reference, the relevant sample configuration for Router ASBR2 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.10.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-3/1/0 {
        unit 0 {
            family inet {
                address 10.0.78.2/30;
            }
            family mpls;
        }
    }
    ge-3/1/1 {
        unit 0 {
            family inet {
                address 10.0.90.13/30;
            }
            family mpls;
        }
    }
}
```

```
routing-options {
    autonomous-system 0.65020;
}
protocols {
    mpls {
        interface ge-3/1/0.0;
        interface ge-3/1/1.0;
    }
    bgp {
        group core-ibgp {
            type internal;
            local-address 192.168.10.1;
            family inet {
                labeled-unicast {
                    resolve-vpn;
                }
            }
            family l2vpn {
                signaling;
            }
            neighbor 192.168.11.1;
        }
        group vpls-metro {
            type external;
            multihop;
            local-address 192.168.10.1;
            family l2vpn {
                signaling;
            }
            peer-as 65010;
            neighbor 192.168.3.1;
        }
        group core-metro {
            type external;
            local-address 10.0.78.2;
            family inet {
                labeled-unicast {
                    resolve-vpn;
                }
            }
            export loopback;
            peer-as 65010;
            neighbor 10.0.78.1;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface ge-3/1/1.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface ge-3/1/0.0;
        interface ge-3/1/1.0;
    }
}
policy-options {
    policy-statement loopback {
        term term1 {
            from {
                protocol [ ospf direct ];
                route-filter 192.168.0.0/16 longer;
            }
            then accept;
        }
    }
}
routing-instances {
    inter-as {
        instance-type vpls;
        route-distinguisher 65020:1;
        vrf-target target:2:1;
        protocols {
            vpls {
                site ASBR-core {
                    site-identifier 2;
                }
                mesh-group core {
                    peer-as {
                        all;
                    }
                }
            }
        }
    }
}
```

10. To display the configuration for Router PE2, use the `show configuration` command.

    For your reference, the relevant sample configuration for Router PE2 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.11.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-0/1/0 {
        unit 0 {
            family inet {
                address 10.0.90.14/30;
            }
            family mpls;
        }
    }
    ge-0/1/1 {
        encapsulation ethernet-vpls;
        unit 0 {
            family vpls;
        }
    }
}
routing-options {
    autonomous-system 0.65020;
}
protocols {
    mpls {
        interface ge-0/1/0.0;
    }
    bgp {
        group core-ibgp {
            type internal;
            local-address 192.168.11.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.168.10.1;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface ge-0/1/0.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        interface ge-0/1/0.0;
    }
}
routing-instances {
    inter-as {
        instance-type vpls;
        interface ge-0/1/1.0;
        route-distinguisher 65020:1;
        vrf-target target:2:1;
        protocols {
            vpls {
                site PE2 {
                    site-identifier 3;
                }
            }
        }
    }
}
```

11. To display the configuration for Router CE2, use the `show configuration` command.

For your reference, the relevant sample configuration for Router CE2 follows.

```
interfaces {
    lo0 {
        unit 0 {
            family inet {
                address 192.168.12.1/32 {
                    primary;
                }
                address 127.0.0.1/32;
            }
        }
    }
    ge-0/1/1 {
        unit 0 {
            family inet {
                address 10.10.11.2/24;
            }
        }
    }
}
```

Configuring VPLS and Integrated Routing and Bridging

IN THIS SECTION

- Configuring MAC Address Flooding and Learning for VPLS | **1139**
- Configuring MSTP for VPLS | **1139**
- Configure IRB in the VPLS network on ACX7000 Family | **1139**

Traditional Layer 2 switching environments consist of Layer 2 devices (such as switches) that partition data into broadcast domains. The broadcast domains can be created through physical topologies or logically through virtual local area networks (VLANs). For MX Series routers, you can logically configure broadcast domains within routing instances with the instance-type `virtual-switch`, routing instances with the instance-type `vpls`, or bridging domains. The individual routing instances or bridging domains are differentiated through VLAN identifiers and these instances or domains function much like traditional VLANs.

To configure a VLAN with IRB support, include the following statements:

```
[edit]
routing-instances {
    instance-name {
        instance-type vpls;
        vlan-id id;
        route-distinguisher distinguisher;
        vrf-target target;
        interface interface-name;
        interface interface-name;
        routing-interface interface-name;
    }
}
```

NOTE: The routing instance of instance-type `vpls` is not support on the ACX7000 family. VPLS routing instances on the ACX7000 family is supported with the instance-type of `virtual-switch`. For information on configuring an IRB with an instance-type of `virtual-switch` on the ACX7000, see "Configure IRB in the VPLS network on ACX7000 Family " on page 1139.

In multihomed VPLS configurations, you can configure VPLS to keep a VPLS connection up if only an IRB interface is available by configuring the `irb` option for the `connectivity-type` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level. The `connectivity-type` statement has the `ce` and `irb` options. The `ce` option is the default and specifies that a CE interface is required to maintain the VPLS connection. By default, if only an IRB interface is available, the VPLS connection is brought down.

NOTE: A maximum of 4096 active logical interfaces are supported for a VLAN or on each mesh group in a VPLS routing instance configured for Layer 2 bridging.

For detailed information and configuration instructions on bridging domains and spanning tree protocol, see Junos OS Network Interfaces Library for Routing Devices and Junos OS Routing Protocols Library.

The following sections provide configuration information specific to VPLS in regards to integrated routing and bridging:

Configuring MAC Address Flooding and Learning for VPLS

In a VPLS routing instance or bridge domain, when a frame is received from a CE interface, it is flooded to the other CE interfaces and all of the VE interfaces if the destination MAC address is not learned or if the frame is either broadcast or multicast. If the destination MAC address is learned on another CE device, such a frame is unicasted to the CE interface on which the MAC address is learned. This might not be desirable if the service provider does not want CE devices to communicate with each other directly.

To prevent CE devices from communicating directly include the `no-local-switching` statement at the `[edit bridge-domains` *`bridge-domain-name`*`]` hierarchy level:

```
[edit bridge-domains bridge-domain-name]
no-local-switching;
```

When you include the `no-local-switching` statement, frames arriving on a CE interface are sent to VE or core-facing interfaces only.

NOTE: When you configure the `no-local-switching` command at the `[edit bridge-domains` *`bridge-domain-name`*`]` hierarchy level, it might not prevent multicast traffic from being forwarded between the CE-facing interfaces of the bridge domain. Broadcast, unknown unicast, and known multicast traffic does not exhibit this behavior.

Configuring MSTP for VPLS

When you configure integrated routing and bridging, you might also need to configure the Multiple Spanning Tree Protocol (MSTP). When you configure MSTP on a provider edge (PE) router running VPLS, you must also configure `ethernet-vpls` encapsulation on the customer-facing interfaces. VLAN-based VPLS interface encapsulations are not supported with MSTP.

Configure IRB in the VPLS network on ACX7000 Family

Junos OS Evolved does not support the routing instance with an instance-type of `vpls`. You must configure a routing instance with an instance-type of `virtual-switch` and a protocol of type `vpls`. To configure IRB support in an VPLS network on devices running Junos OS Evolved, include the following statements:

```
set routing-instances routing-instance-name instance-type virtual-switch
set routing-instances routing-instance-name protocols vpls neighbor ip-address
set routing-instances routing-instance-name protocols vpls no-tunnel-services
set routing-instances routing-instance-name protocols vpls vpls-id number
set routing-instances routing-instance-name vlans vlan-name vlan-id number
set routing-instances routing-instance-name vlans vlan-name interface interface-name
set  routing-instances routing-instance-name l3-interface interface-name
set interfaces interface-name unit number family inet address ip-address arp ip-address mac mac-
address
```

The following sample configuration shows a routing instance with an instance-type of `virtual-switch`on an IRB interface:

```
user@router# show routing-instances
green0 {
    instance-type virtual-switch;
    protocols {
        vpls {
            neighbor 192.168.1.1;
            control-word;
            no-tunnel-services;
            vpls-id 200;
        }
    }
    vlans {
        bd1 {
            vlan-id 1;
            interface et-1/0/0.0;
            interface et-1/0/1.0;
            l3-interface irb.0;
        }
    }
}
user@router# show interfaces irb
unit 0 {
    family inet {
        address 10.1.1.1/24 {
            arp 10.1.1.10 mac 00:00:05:00:53:01;
        }
    }
}
```

RELATED DOCUMENTATION

Configuring Integrated Routing and Bridging for VLANs

Configuring Integrated Routing and Bridging in a VPLS Instance (MX Series Routers Only) | **1141**

Example: Configuring Inter-AS VPLS with MAC Processing at the ASBR | **1101**

Example: VPLS Configuration (BGP and LDP Interworking) | **653**

Example: VPLS Configuration (BGP Signaling) | **637**

Configuring Integrated Routing and Bridging in a VPLS Instance (MX Series Routers Only)

Integrated routing and bridging (IRB) over VPLS cannot be used in conjunction with the `vlan-id all` statement. One or more Layer 2 logical interfaces must be configured inside the instance in order for IRB to function properly.

To configure IRB within a VPLS instance, include the `routing-interface` *`irb-interface-name`* statement at the [`edit routing-instances` *`routing-instance-name`* `instance-type vpls`] hierarchy level:

```
[edit]
routing-instances {
    marketing {
        instance-type vpls;
        vlan-id 123;
        route-distinguisher 11.11.11.11:10;
        vrf-target target:100:100;
        interface ae0.100;
        interface ae0.200;
        routing-interface irb.1234;
    }
}
```

RELATED DOCUMENTATION

Configuring Integrated Routing and Bridging for VLANs

Configuring VPLS and Integrated Routing and Bridging | **1137**

Example: Configuring Inter-AS VPLS with MAC Processing at the ASBR | **1101**

Example: VPLS Configuration (BGP and LDP Interworking) | **653**

Example: VPLS Configuration (BGP Signaling) | 637

Configuring L2PT for VPLS

You can use Layer 2 Protocol Tunneling (L2PT) to transparently send packets across a VPLS network. The customer edge (CE) device sends and receives packets over VPLS pseudowires to and from the provider edge (PE) device. L2PT doesn't interfere with protocol instances in the network. L2PT supports 802.1x, 802.3ah, CDP, E-LMI, MVRP, LACP, STP/RSTP/MSTP, LLDP, MMRP, and VTP Layer 2 control protocols.

Use Feature Explorer to determine if a feature is supported on your platform and release.

Here is how to configure L2PT for VPLS on the PE device.

1. The packets leave the CE from an untagged or single-tagged port called `interface` *`CE-interface-name`*. Configure VPLS VLAN encapsulation for packets that arrive from this port.

   a. Enable flexible VLAN tagging and flexible Ethernet services so you can configure more than one instance of VPLS encapsulation on the port.

   ```
   [edit interfaces CE-interface-name]
   set flexible-vlan-tagging
   set encapsulation flexible-ethernet-services
   ```

   b. Since the L2CP frames from the CE device are untagged, configure the native VLAN ID to support untagged VLAN packets.

   ```
   [edit interfaces CE-interface-name]
   set native-vlan-id vlan-id-number
   ```

   c. Configure VPLS VLAN encapsulation.

   ```
   [edit interfaces CE-interface-name unit unit-number]
   set encapsulation vlan-vpls
   set vlan-id vlan-id-number
   ```

2. Configure the routing instance.

a. Configure VPLS as the routing instance protocol.

```
[edit routing-instances routing-instance-name]
set protocols vpls
```

b. (Optional) If your router doesn't have a Tunnel Services PIC, you will need to configure the `no-tunnel-services` statement at the `[edit routing-instances routing-instance-name protocols vpls]` hierarchy level.

c. Configure the instance type and identify the CE port. Use the VLAN ID from Step 1.

```
[edit routing-instances routing-instance-name]
set instance-type vpls
set vlan-id vlan-id-number
set interface CE-interface-name
```

3. Configure the L2PT protocol. The protocol can be any of the following: `(stp | vtp | cdp | ieee8021x | ieee8023ah | elmi | lacp | lldp | mmrp | mvrp)`.

```
[edit protocols layer2-control mac-rewrite interface interface-name]
set protocol protocol-name
```

You have configured L2PT for VPLS on your network.

Platform Specific L2PT Behavior

(ACX Series) These devices don't support L2PT on logical tunnel, access, or trunk interfaces.

(ACX Series) When L2PT is configured for a protocol, the `drop-threshold`, `shutdown-threshold`, and class-of-service parameters can't be configured for that protocol on the same interface.

(ACX Series) If any L2PT supported L2 control protocol is configured on an interface, then you can't configure the same L2PT protocol again on that interface. The CLI will show an error and the configuration won't commit. Due to this restriction, LACP for L2PT is not supported on aggregated ethernet (AE) interfaces, either static or dynamic type. This limitation only applies to user-to-network (UNI) AE interfaces. L2PT can co-exist with L2 control protocol configuration on network-to-network (NNI) AE interfaces, and non-AE UNI interfaces.

RELATED DOCUMENTATION

Layer 2 Protocol Tunneling

CHAPTER 39

Configuring Load Balancing and Performance

IN THIS CHAPTER

- Configuring VPLS Load Balancing | **1146**
- Configuring VPLS Load Balancing Based on IP and MPLS Information | **1148**
- Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms | **1150**
- Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves | **1152**
- Understanding MAC Pinning | **1174**
- Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**
- Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**
- Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**
- Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**
- Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**
- Configuring MAC Pinning on VPLS CE Interface | **1185**
- Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance | **1187**
- Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance | **1189**
- Configuring MAC Pinning on Access Interfaces for Logical Systems | **1191**
- Configuring MAC Pinning on Trunk Interfaces for Logical Systems | **1193**
- Configuring MAC Pinning on Access Interfaces in Virtual Switches for Logical Systems | **1195**
- Configuring MAC Pinning on Trunk Interfaces in Virtual Switches for Logical Systems | **1198**
- Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) for Logical Systems | **1200**
- Configuring MAC Pinning on VPLS CE Interface for Logical Systems | **1203**
- Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance for Logical Systems | **1205**
- Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance for Logical Systems | **1207**
- Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Access Interfaces | **1209**

- Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Trunk Interfaces | **1215**
- Configuring Improved VPLS MAC Address Learning on T4000 Routers with Type 5 FPCs | **1225**
- Understanding Qualified MAC Learning | **1227**
- Qualified Learning VPLS Routing Instance Behavior | **1228**
- Configuring Qualified MAC Learning | **1235**

Configuring VPLS Load Balancing

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

You can configure the Junos OS so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This feature is called per-packet load balancing. You can use load balancing to spread traffic across multiple paths between routers. You can also configure per-packet load balancing to optimize VPLS traffic flows across multiple paths.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

You can load-balance VPLS traffic based on Layer 2 media access control (MAC) information, IP information and MPLS labels, or MPLS labels only.

NOTE: For platform support information, see *family multiservice*.

To optimize VPLS traffic flows across multiple paths, include the `family multiservice` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
family multiservice {
    destination-mac;
    label-1;
    label-2;
    payload {
        ip {
            layer-3 {
                (destination-ip-only | source-ip-only);
            }
            layer-3-only;
            layer-4;
        }
    }
    source-mac;
    symetric-hash {
        complement;
    }
}
```

You can configure one or more of the following options to load-balance using the specified packet information:

- `destination-mac`—Include the destination-address MAC information in the hash key for Layer 2 load balancing.
- `source-mac`—Include the source-address MAC information in the hash key.
- `label-1`—Include the first MPLS label in the hash key. Used for including a one-label packet for per-flow load balancing of IPv4 VPLS traffic based on IP information and MPLS labels.
- `label-2` —Include the second MPLS label in the hash key. If both `label-1` and `label-2` are specified, the entire first label and the first 16 bits of the second label are hashed.
- `payload`—Include the packet's IP payload in the hash key.
- `ip`—Include the IP address of the IPv4 or IPv6 payload in the hash key.
- `layer-3-only`—Include only the Layer 3 information from the packet's IP payload in the hash key.
- `layer-3`—Include Layer 3 information from the packet's IP payload in the hash key.
- `destination-address-only`—Include only the destination IP address in the payload in the hash key.

  NOTE: You can include either the `source-address-only` or the `destination-address-only` statement, not both. They are mutually exclusive.

- `source-address-only`—Include only the source IP address in the payload in the hash key.

NOTE: You can include either the `source-address-only` or the `destination-address-only` statement, not both. They are mutually exclusive.

- `layer-4`—Include Layer 4 information from the packet's IP payload in the hash key.
- `symmetric-hash`—Configure the symmetric hash or symmetric hash complement for configuring symmetrical load balancing on an 802.3ad Link Aggregation Group.
- `complement`—Include the complement of the symmetric hash in the hash key.

For more information about how to configure per-packet load balancing, see theRouting Policies, Firewall Filters, and Traffic Policers User Guide.

RELATED DOCUMENTATION

Configuring VPLS Load Balancing Based on IP and MPLS Information | **1148**

Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms | **1150**

Configuring VPLS Load Balancing Based on IP and MPLS Information

Load balancing for VPLS traffic is handled differently by different routing platforms. For more information, see "Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms" on page 1150.

For IPv4 traffic, only the IP source and destination addresses are included in the hash key. For MPLS and IPv4 traffic, one or two MPLS labels and IPv4 source and destination addresses are included. For MPLS Ethernet pseudowires, only one or two MPLS labels are included in the hash key.

To optimize VPLS flows across multiple paths based on IP and MPLS information, include the `family multiservice` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
[edit forwarding-options hash-key]
family multiservice {
    label-1;
    label-2;
    payload {
        ip {
            layer-3-only;
        }
    }
}
```

To use the first MPLS label in the hash key, include the `label-1` statement:

```
[edit forwarding-options hash-key family multiservice]
label-1;
```

To use the second MPLS label, include both the `label-1` and `label-2` statements:

```
[edit forwarding-options hash-key family multiservice]
label-1;
label-2;
```

To use the packet's IPv4 payload in the hash key, include the `payload` and `ip` statements:

```
[edit forwarding-options hash-key family multiservice]
payload {
    ip;
}
```

NOTE: Only IPv4 is supported.

To include only Layer 3 information from the IPv4 payload, specify the `layer-3-only` option to the `payload ip` statement:

```
[edit forwarding-options hash-key family multiservice]
payload {
    ip {
        layer-3-only;
    }
}
```

To use the first and second MPLS labels and the packet's IP payload in the hash key, include the `label-1`, `label-2`, and `payload ip` statements:

```
[edit forwarding-options hash-key family multiservice]
label-1;
label-2;
payload {
    ip;
}
```

Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms

In Junos OS Release 9.5 and later, on MX Series routers, you can configure the load balancing hash key for Layer 2 traffic to use fields in the Layer 3 and Layer 4 headers inside the frame payload. You can also configure VPLS load balancing based on IP and MPLS information on M120 and M320 routers only. For more information, see "Configuring VPLS Load Balancing Based on IP and MPLS Information" on page 1148.

You can configure load balancing on MX Series routers based on Layer 3 or Layer 4 information or both.

To configure VPLS load balancing on the MX Series router to include either Layer 3 IP information or Layer 4 headers or both:

1. Include the `payload` statement at the `[edit forwarding-options hash-key family multiservice]` hierarchy level.
2. Include the `ip` statement at the `[edit forwarding-options hash-key family multiservice payload]` hierarchy level.

To configure VPLS load balancing to include the Layer 3 information:

1. Include the `layer-3` statement at the `[edit forwarding-options hash-key family multiservice payload ip]` hierarchy level.

2. Include the `source-ip-only` statement at the `[edit forwarding-options hash-key family multiservice payload ip layer-3]` hierarchy level to include information about the IP source address only in the hash key.

3. Include `destination-ip-only` statement at the `[edit forwarding-options hash-key family multiservice payload ip layer-3]` hierarchy level to include information about the IP destination address only in the hash key.

NOTE: You can configure either the `source-ip-only` or the `destination-ip-only` statements at a time, not both. They are mutually exclusive.

To configure VPLS load balancing to include Layer 4 information:

- Include the `layer-4` statement at the `[edit forwarding-options hash-key family multiservice payload ip]` hierarchy level.

The following example shows load balancing configured to use the source Layer 3 IP address option and Layer 4 header fields as well as the source and destination MAC addresses:

```
[edit forwarding-options hash-key]
family multiservice {
    source-mac;
    destination-mac;
    payload {
        ip {
            layer-3 {
                source-ip-only;
            }
            layer-4;
        }
    }
}
```

RELATED DOCUMENTATION

family multiservice

hash-key

Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves

IN THIS SECTION

- MAC Moves Loop Prevention in VPLS Network Overview | **1152**
- Configuring VPLS Loop Prevention Due to MAC Moves | **1154**
- Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves | **1156**

MAC Moves Loop Prevention in VPLS Network Overview

Starting in Junos OS 14.2, you can configure the router to prevent a loop in a VPLS network. In a virtual private LAN service (VPLS) deployment, when a previously learned media access control (MAC) address appears on a different physical interface, for example, local interfaces (Gigabit Ethernet interfaces) or label switched Interfaces (LSIs), or within a different unit of the same physical interface and if this behavior occurs frequently, then it is considered a MAC move.

You can configure the router to report a MAC address move based on the following parameters:

- Number of times a MAC address move occurs
- Specified period of time over which the MAC address move occurs

Configuration errors at the network can force traffic into never ending circular paths or loops. These loops in the VPLS network cause frequent MAC moves between different interfaces which can be used to rectify the problem by disabling such an interface in the network. The following two approaches can be used to disable the interface causing the loop:

- Base learning interface (base IFL) approach algorithm- This is the primary approach used to disable the looped interface. Base interface information is maintained for every MAC in the routing instance. If the MAC stays at the interface it was first learned for 300 seconds, then the interface-MAC association is considered to be stable and this interface is considered as the base interface of the MAC. If the MAC move happens frequently between the local interface and the LSI interface while the base interface of the MAC is an LSI, then the local interface is considered to be looped and has to be disabled.
- Statistical approach algorithm– This is the secondary approach used to disable the looped interface. If the MAC has not been learned for over 300 seconds at an interface, then it does not have a base interface and hence the statistical approach is used.

If the MAC that has no base interface information(Base Learning interface is null) starts moving, then the statistics of such MAC moves between different interfaces is learned. If the statistics show MAC moves from LSI to local interface or from local interface to local interface, then the local interface is considered to be looped and is disabled.

There are certain MACs that can move between different interfaces, for example, primary role change in the Virtual Router Redundancy Protocol (VRRP). The base interface of such MAC moves cannot be maintained as this leads to the assumption of a loop creation. Hence, such MACs should be configured as virtual MACs. Example of virtual MACs are 00:00:5e:00:01:xx for VRRP, 00:00:0c:07:ac:xx for hot standby router protocol (HSRP) , 00:07:b4:00:01:xx for global server load balancing (GSLB), and 02:bf:xx:xx:xx:xx for VMotion.

Starting with Junos OS Release 17.4R1, the `global-mac-move` statement replaces the `vpls-mac-move` statement. The following timers under the `global-mac-move` statement help in monitoring the disabled interfaces:

- Cooloff time — The cooloff time starts when the interface gets disabled. During this time any MAC move happening in the routing instance is ignored. This ensures that only one interface is blocked at a given time on a routing instance, and blocking of another interface happens only after the expiration of the cooloff timer provided the given MAC moves are still observed. By default, the cooloff time is 30 seconds.

- Interface recovery time — When an interface gets disabled , it is disabled permanently. Configuring the interface recovery time ensures that the interface gets enabled on completion of the interface recovery time duration. We recommend that you configure an interface recovery time of more than 300 seconds.

- Statistical approach wait time — The time when the statistics are collected after MAC moves are observed to determine the existence of a loop when there is no base IFL for the MAC address. . By default, the statistical approach wait timer is 30 seconds.

  Before the base learning interface of a MAC address is established, the statistical approach algorithm is used in MAC move loop prevention. When a statistical approach algorithm is used, the offending MAC address is shown with a MAC address of 00:00:00:00:00:00. Until the base learning interface of MAC addresses are established, this may cause interfaces with routing loops to be misidentified.

SEE ALSO

virtual-mac

global-mac-move

Configuring VPLS Loop Prevention Due to MAC Moves

In a virtual private LAN service (VPLS) deployment, when a previously learned media access control (MAC) address appears on a different physical interface, for example, local interfaces (Gigabit Ethernet interfaces) or label switched Interfaces (LSIs), or within a different unit of the same physical interface and if this behavior occurs frequently, then it is considered a MAC move. The router reports a MAC address move based on the number of times a MAC address move occurs and the specified period of time over which the MAC address move occurs. Configuration errors at the VPLS network can lead to loops that cause frequent MAC moves between different interfaces. These moves can be used to rectify the problem by disabling such interface in the network. The following two approaches can be used to disable the interface:

- Base learning interface (base IFL) approach algorithm — This is the primary approach used to disable the looped local interface.
- Statistical approach algorithm — This is the secondary approach used to disable the looped local interface.

Some virtual MACs can genuinely move between different interfaces and such MACs can be configured to ignore the moves. The cooloff time and the statistical approach wait time are used internally to find out the looped interface. The interface recovery time can be configured to auto-enable the interface that gets disabled due to a loop in the network.

Before you begin to configure loop prevention in a VPLS network:

1. Configure the VPLS topology.
2. Configure the VPLS routing instances.
3. Enable VPLS MAC move action on a VPLS instance.
4. Configure the routing and signaling protocols.

The following uses the `global-mac-move` command, which replaced the `vpls-mac-move` command starting with Junos OS Release 17.4R1.

To configure loop prevention in a VPLS network:

1. Configure the threshold time and the threshold count to detect MAC moves.

```
[edit protocols l2-learning global-mac-move]
user@host# set threshold-time seconds
user@host# set threshold-count seconds
```

For example, configure the threshold time as 30 and the threshold count as 4 to detect MAC moves.

```
[edit protocols l2-learning global-mac-move]
user@host# set threshold-time 30
user@host# set threshold-count 4
```

NOTE: If the threshold time and threshold count are not configured, then the default values are used. The default value of threshold time is 1 second, and default value of threshold count is 50.

2. (Optional) Configure a cooloff time to ensure that no other interface gets disabled once an interface, for a routing instance, is disabled during this time period.

```
[edit protocols l2-learning global-mac-move]
user@host# set cooloff-time seconds
```

3. (Optional) Configure the statistical approach wait time to determine the existence of a loop based on the statistics collected after MAC moves are observed when there is no base IFL for the MAC address.

```
[edit protocols l2-learning global-mac-move]
user@host# set statistical-approach-wait-time seconds
```

4. (Optional) Configure the interface recovery time to ensure that the disabled interface gets enabled on completion of the interface recovery time duration.

```
[edit protocols l2-learning global-mac-move]
user@host# set interface-recovery-time seconds
```

5. (Optional) Configure the virtual MAC address to ignore the MAC moves as this leads to the assumption of loop creation.

```
[edit protocols l2-learning global-mac-move]
user@host# set virtual-mac mac-address
```

6. Configure the VPLS routing instance of an interface.

```
[edit routing-instances instance-name]
user@host# set instance-type vpls
user@host# set interface interface
```

7. Enable MAC move action on the interface for the VPLS instance, and configure the VPLS neighbor.

```
[edit routing-instances instance-name]
user@host# set protocols vpls enable-mac-move-action
user@host# set protocols vpls no-tunnel-services
user@host# set protocols vpls vpls-id vpls-id
user@host# set protocols vpls neighbor IP-address
```

SEE ALSO

Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves | **1152**

Example: Configuring Loop Prevention in VPLS Network Due to MAC Moves

IN THIS SECTION

- Requirements | **1156**
- Overview | **1157**
- Configuration | **1158**
- Verification | **1168**

This example shows how to prevent a loop in the VPLS network due to MAC moves between different physical interfaces.

When a MAC move is detected in the VPLS network, Junos OS allows the prevention of the loop in the network by disabling the interface using a base IFL or statistical approach algorithm.

Requirements

This example uses the following hardware and software components:

- Four MX Series 5G Universal Routing Platforms
- Junos OS Release 14.2 or later running on all devices

Before you begin:

1. Configure the VPLS topology.
2. Configure the VPLS routing instances.
3. Enable VPLS MAC move action on a VPLS instance.
4. Configure the routing and signaling protocols.

Overview

IN THIS SECTION

● Topology | 1157

Starting with Junos OS Release 14.2, the loop creation in the VPLS network due to frequent MAC moves between different physical interfaces can be prevented by identifying and disabling such interfaces using the base IFL approach or statistical approach algorithm.

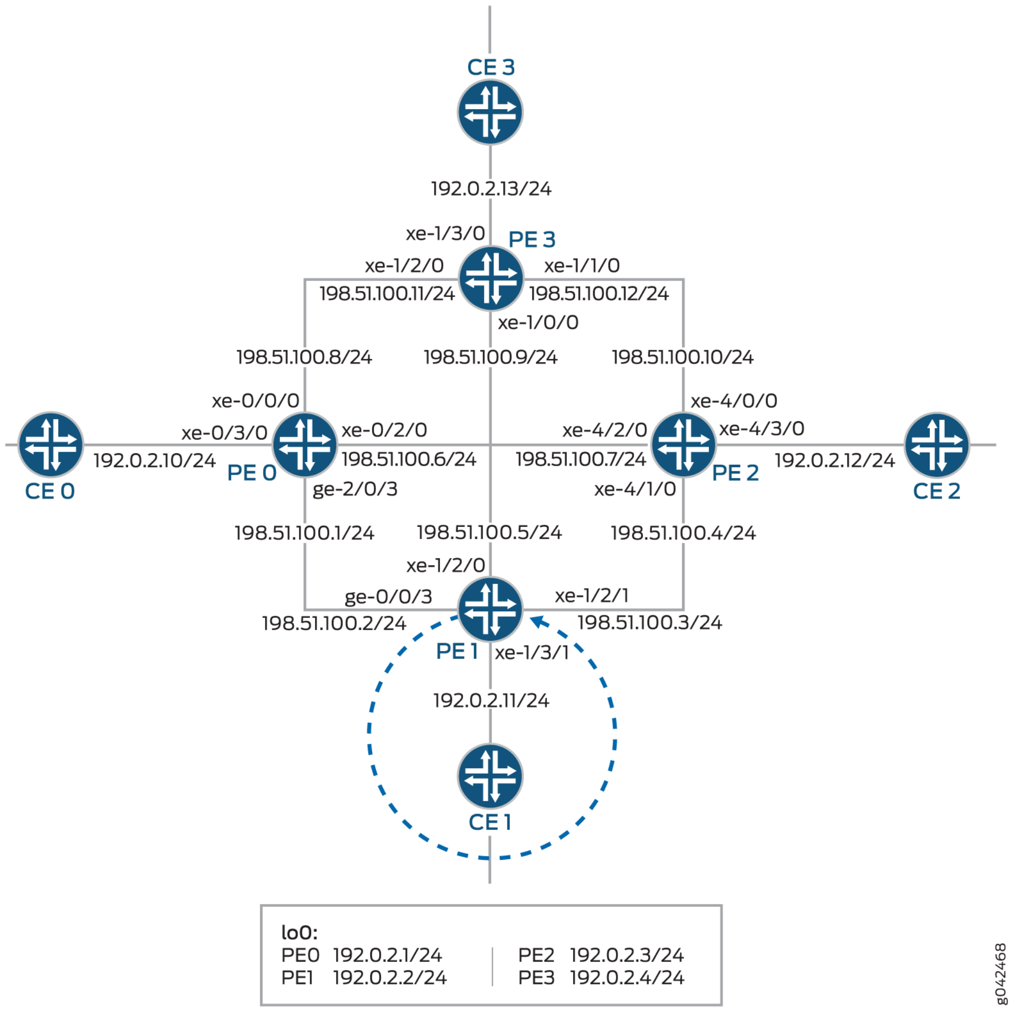
The *base IFL approach* algorithm is the primary approach. Base interface information is maintained for every MAC in the routing instance. If the MAC stays at the interface it was first learned for 300 seconds, then the interface-MAC association is considered to be stable, and this interface is considered as the base interface of the MAC. If the MAC move happens frequently between the local interface and the LSI interface while the base interface of the MAC is an LSI, then the local interface is considered to be looped and has to be disabled.

The *statistical approach* algorithm is the secondary approach used to disable the looped interface. If the MAC has not been learned for over 300 seconds at an interface, then it does not have a base interface and hence the statistical approach is used. If the MAC that has no base interface information(Base Learning interface is null) starts moving then the statistics of such MAC moves between different interfaces is learned. If the statistics show MAC moves from LSI to local interface or from local interface to local interface then the local interface is considered to be looped and is disabled.

Topology

In the topology shown in Figure 84 on page 1158, a loop was detected on Device PE1 in the VPLS network.

Figure 84: Example Loop Prevention Due to MAC Move in VPLS Network



Configuration

IN THIS SECTION

- CLI Quick Configuration | 1159
- Configuring Device PE1 | 1163
- Results | 1166

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.

This example uses the `global-mac-move` command, which replaced the `vpls-mac-move` command starting with Junos OS Release 17.4R1.

PE0

```
set interfaces xe-0/0/0 unit 0 family inet address 198.51.100.8/24
set interfaces xe-0/0/0 unit 0 family mpls
set interfaces xe-0/2/0 unit 0 family inet address 198.51.100.6/24
set interfaces xe-0/2/0 unit 0 family mpls
set interfaces xe-0/3/0 vlan-tagging
set interfaces xe-0/3/0 encapsulation vlan-vpls
set interfaces xe-0/3/0 unit 600 encapsulation vlan-vpls
set interfaces xe-0/3/0 unit 600 vlan-id 600
set interfaces ge-2/0/3 unit 0 family inet address 198.51.100.1/24
set interfaces ge-2/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 192.0.2.1/24
set routing-options router-id 192.0.2.1
set routing-options autonomous-system 701
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-2/0/3.0
set protocols mpls interface xe-0/2/0.0
set protocols mpls interface xe-0/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface ge-2/0/3.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface ge-2/0/3.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-0/2/0.0 interface-type p2p
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-0/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-0/0/0.0 metric 10
set protocols ldp interface xe-0/0/0.0
set protocols ldp interface xe-0/2/0.0
set protocols ldp interface ge-2/0/3.0
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set protocols l2-learning global-mac-move threshold-time 30
set protocols l2-learning global-mac-move threshold-count 4
set protocols l2-learning global-mac-move cooloff-time 10
set protocols l2-learning global-mac-move statistical-approach-wait-time 10
set protocols l2-learning global-mac-move interface-recovery-time 5
set protocols l2-learning global-mac-move virtual-mac 00:00:5e:00:01:00/40
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface xe-0/3/0.600
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls vpls-id 100
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.2
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.3
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.4
```

PE1

```
set interfaces ge-0/0/3 unit 0 family inet address 198.51.100.2/24
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces xe-1/2/0 unit 0 family inet address 198.51.100.5/24
set interfaces xe-1/2/0 unit 0 family mpls
set interfaces xe-1/2/1 unit 0 family inet address 198.51.100.3/24
set interfaces xe-1/2/1 unit 0 family mpls
set interfaces xe-1/3/1 vlan-tagging
set interfaces xe-1/3/1 encapsulation vlan-vpls
set interfaces xe-1/3/1 unit 600 encapsulation vlan-vpls
set interfaces xe-1/3/1 unit 600 vlan-id 600
set interfaces lo0 unit 0 family inet address 192.0.2.2/24
set routing-options router-id 192.0.2.2
set routing-options autonomous-system 701
set protocols mpls interface fxp0.0 disable
set protocols mpls interface ge-0/0/3.0
set protocols mpls interface xe-1/2/1.0
set protocols mpls interface xe-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface ge-0/0/3.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-1/2/1.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-1/2/1.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-1/2/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-1/2/0.0 metric 10
set protocols ldp interface ge-0/0/3.0
set protocols ldp interface xe-1/2/0.0
set protocols ldp interface xe-1/2/1.0
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set protocols l2-learning global-mac-move threshold-time 30
set protocols l2-learning global-mac-move threshold-count 4
set protocols l2-learning global-mac-move cooloff-time 10
set protocols l2-learning global-mac-move statistical-approach-wait-time 10
set protocols l2-learning global-mac-move interface-recovery-time 10
set protocols l2-learning global-mac-move virtual-mac 00:00:5e:00:01:00/40
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface xe-1/3/1.600
set routing-instances vpls_1 protocols vpls enable-mac-move-action
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls vpls-id 100
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.1
```

PE2

```
set interfaces xe-4/0/0 unit 0 family inet address 198.51.100.10/24
set interfaces xe-4/0/0 unit 0 family mpls
set interfaces xe-4/1/0 unit 0 family inet address 198.51.100.4/24
set interfaces xe-4/1/0 unit 0 family mpls
set interfaces xe-4/2/0 unit 0 family inet address 198.51.100.7/24
set interfaces xe-4/2/0 unit 0 family mpls
set interfaces xe-4/3/0 vlan-tagging
set interfaces xe-4/3/0 encapsulation vlan-vpls
set interfaces xe-4/3/0 unit 600 encapsulation vlan-vpls
set interfaces xe-4/3/0 unit 600 vlan-id 600
set interfaces lo0 unit 0 family inet address 192.0.2.3/24
set routing-options router-id 192.0.2.3
set routing-options autonomous-system 701
set protocols mpls interface fxp0.0 disable
set protocols mpls interface xe-4/2/0.0
set protocols mpls interface xe-4/1/0.0
set protocols mpls interface xe-4/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface xe-4/2/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-4/2/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-4/1/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-4/1/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-4/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-4/0/0.0 metric 10
set protocols ldp interface xe-4/0/0.0
set protocols ldp interface xe-4/1/0.0
set protocols ldp interface xe-4/2/0.0
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set protocols l2-learning global-mac-move threshold-time 30
set protocols l2-learning global-mac-move threshold-count 4
set protocols l2-learning global-mac-move threshold-count 5
set protocols l2-learning global-mac-move cooloff-time 10
set protocols l2-learning global-mac-move statistical-approach-wait-time 10
set protocols l2-learning global-mac-move interface-recovery-time 10
set protocols l2-learning global-mac-move virtual-mac 00:00:5e:00:01:00/40
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface xe-4/3/0.600
set routing-instances vpls_1 protocols vpls enable-mac-move-action
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls vpls-id 100
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.1
```

PE3

```
set interfaces xe-1/0/0 unit 0 family inet address 198.51.100.9/24
set interfaces xe-1/0/0 unit 0 family mpls
set interfaces xe-1/1/0 unit 0 family inet address 198.51.100.12/24
set interfaces xe-1/1/0 unit 0 family mpls
set interfaces xe-1/2/0 unit 0 family inet address 198.51.100.11/24
set interfaces xe-1/2/0 unit 0 family mpls
set interfaces xe-1/3/0 vlan-tagging
set interfaces xe-1/3/0 encapsulation vlan-vpls
set interfaces xe-1/3/0 unit 600 encapsulation vlan-vpls
set interfaces xe-1/3/0 unit 600 vlan-id 600
set interfaces xe-2/3/0 vlan-tagging
set interfaces xe-2/3/0 encapsulation vlan-vpls
set interfaces xe-2/3/0 unit 600 encapsulation vlan-vpls
set interfaces xe-2/3/0 unit 600 vlan-id 600
set interfaces lo0 unit 0 family inet address 192.0.2.4/24
set routing-options router-id 192.0.2.4
set routing-options autonomous-system 701
set protocols mpls interface fxp0.0 disable
set protocols mpls interface xe-1/2/0.0
set protocols mpls interface xe-1/0/0.0
set protocols mpls interface xe-1/1/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface xe-1/2/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-1/2/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-1/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-1/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface xe-1/1/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface xe-1/1/0.0 metric 10
set protocols ldp interface xe-1/0/0.0
set protocols ldp interface xe-1/1/0.0
set protocols ldp interface xe-1/2/0.0
set protocols ldp interface fxp0.0 disable
set protocols ldp interface lo0.0
set protocols l2-learning global-mac-move threshold-time 30
set protocols l2-learning global-mac-move threshold-count 4
set protocols l2-learning global-mac-move cooloff-time 10
set protocols l2-learning global-mac-move statistical-approach-wait-time 10
set protocols l2-learning global-mac-move interface-recovery-time 10
set protocols l2-learning global-mac-move virtual-mac 00:00:52:00:01:00/40
set policy-options policy-statement pplb then load-balance per-packet
set routing-instances vpls_1 instance-type vpls
set routing-instances vpls_1 interface xe-1/3/0.600
set routing-instances vpls_1 interface xe-2/3/0.600
set routing-instances vpls_1 protocols vpls no-tunnel-services
set routing-instances vpls_1 protocols vpls vpls-id 100
set routing-instances vpls_1 protocols vpls neighbor 192.0.2.1
```

Configuring Device PE1

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 PE1:

NOTE: Repeat this procedure for Routers PE0, PE2, and PE3 after modifying the appropriate interface names, addresses, and any other parameters for the router.

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-0/0/3 unit 0 family inet address 198.51.100.2/24
user@PE1# set ge-0/0/3 unit 0 family mpls
user@PE1# set xe-1/2/0 unit 0 family inet address 198.51.100.5/24
user@PE1# set xe-1/2/0 unit 0 family mpls
user@PE1# set xe-1/2/1 unit 0 family inet address 198.51.100.3/24
user@PE1# set xe-1/2/1 unit 0 family mpls
user@PE1# set xe-1/3/1 vlan-tagging
user@PE1# set xe-1/3/1 encapsulation vlan-vpls
user@PE1# set xe-1/3/1 unit 600 encapsulation vlan-vpls
user@PE1# set xe-1/3/1 unit 600 vlan-id 600
user@PE1# set lo0 unit 0 family inet address 192.0.2.2/24
```

2. Configure the routing options.

```
[edit routing-options]
user@PE1# set router-id 192.0.2.2
user@PE1# set autonomous-system 701
```

3. Configure the MPLS protocol on the interface.

```
[edit protocols mpls]
user@PE1# set interface fxp0.0 disable
user@PE1# set interface ge-0/0/3.0
user@PE1# set interface xe-1/2/1.0
user@PE1# set interface xe-1/2/0.0
```

4. Configure the OSPF protocol.

```
[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface lo0.0 passive
user@PE1# set area 0.0.0.0 interface fxp0.0 disable
user@PE1# set area 0.0.0.0 interface ge-0/0/3.0 interface-type p2p
user@PE1# set area 0.0.0.0 interface ge-0/0/3.0 metric 10
user@PE1# set area 0.0.0.0 interface xe-1/2/1.0 interface-type p2p
user@PE1# set area 0.0.0.0 interface xe-1/2/1.0 metric 10
```

```
user@PE1# set area 0.0.0.0 interface xe-1/2/0.0 interface-type p2p
user@PE1# set area 0.0.0.0 interface xe-1/2/0.0 metric 10
```

5. Configure the LDP protocols on the interfaces.

```
[edit protocols ldp]
user@PE1# set interface ge-0/0/3.0
user@PE1# set interface xe-1/2/0.0
user@PE1# set interface xe-1/2/1.0
user@PE1# set interface fxp0.0 disable
user@PE1# set interface lo0.0
```

6. Configure the threshold time and the threshold count to detect MAC moves.

```
[edit protocols l2-learning global-mac-move]
user@PE1# set threshold-time 30
user@PE1# set threshold-count 4
```

7. Configure VPLS MAC move parameters like cooloff time, statistical approach wait time, interface recovery time, and virtual MAC.

```
[edit protocols l2-learning global-mac-move]
user@PE1# set cooloff-time 10
user@PE1# set statistical-approach-wait-time 10
user@PE1# set interface-recovery-time 10
user@PE1# set virtual-mac 00:00:5e:00:01:00/40
```

8. Enable MAC move action on the interface for the VPLS instance vpls_1.

```
[edit routing-instances vpls_1]
user@PE1# set instance-type vpls
user@PE1# set interface xe-1/3/1.600
user@PE1# set protocols vpls enable-mac-move-action
user@PE1# set protocols vpls no-tunnel-services
user@PE1# set protocols vpls vpls-id 100
user@PE1# set protocols vpls neighbor 192.0.2.1
```

Results

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

```
user@PE1# show interfaces
ge-0/0/3 {
    unit 0 {
        family inet {
            address 198.51.100.2/24;
        }
        family mpls;
    }
}
xe-1/2/0 {
    unit 0 {
        family inet {
            address 198.51.100.5/24;
        }
        family mpls;
    }
}
xe-1/2/1 {
    unit 0 {
        family inet {
            address 198.51.100.3/24;
        }
        family mpls;
    }
}
xe-1/3/1 {
    vlan-tagging;
    encapsulation vlan-vpls;
    unit 600 {
        encapsulation vlan-vpls;
        vlan-id 600;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.2/24;
        }
    }
}
```

```
user@PE1# show protocols
mpls {
    interface fxp0.0 {
        disable;
    }
    interface ge-0/0/3.0;
    interface xe-1/2/1.0;
    interface xe-1/2/0.0;
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fxp0.0 {
            disable;
        }
        interface ge-0/0/3.0 {
            interface-type p2p;
            metric 10;
        }
        interface xe-1/2/1.0 {
            interface-type p2p;
            metric 10;
        }
        interface xe-1/2/0.0 {
            interface-type p2p;
            metric 10;
        }
    }
}
ldp {
    interface ge-0/0/3.0;
    interface xe-1/2/0.0;
    interface xe-1/2/1.0;
    interface fxp0.0 {
        disable;
    }
    interface lo0.0;
}
l2-learning {
    global-mac-move {
        threshold-time 30;
        threshold-count 4;
    }
    global-mac-move {
        cooloff-time 10;
        statistical-approach-wait-time 10;
        interface-recovery-time 10;
        virtual-mac 00:00:5e:00:01:00/40;
    }
}
```

```
user@PE1# show routing-instances
vpls_1 {
    instance-type vpls;
    interface xe-1/3/1.600;
    protocols {
        vpls {
            enable-mac-move-action;
            no-tunnel-services;
            vpls-id 100;
            neighbor 192.0.2.1;
        }
    }
}
```

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

Verification

IN THIS SECTION

- Verifying the MAC Move in a VPLS Network | **1169**
- Verifying the MAC Move in a VPLS Instance | **1169**

- Verifying the MAC Move Buffer in a VPLS Network | 1170
- Verifying the VPLS MAC Table for the Base IFL Approach Algorithm | 1171
- Verifying That the Interface Is Disabled | 1172
- Verifying the VPLS MAC Table for the Statistical Approach Algorithm | 1173

Verify that the configuration is working properly.

Verifying the MAC Move in a VPLS Network

Purpose

Verify that the MAC move is observed in a VPLS network.

Action

From operational mode, run the `show vpls mac-move-action` command for Device PE1.

```
user@PE1> show vpls mac-move-action
Instance: vpls_1
   Local interface: xe-1/3/1.600, Index: 341
     Algorithm used : Base IFL
     Time rec       : 02:30:35
     Recovery timer : Yes
```

Meaning

The output shows Instance name, Local interface and Algorithm used indicating that VPLS MAC Move is observed in a VPLS network.

Verifying the MAC Move in a VPLS Instance

Purpose

Verify that the MAC move is observed in a VPLS instance.

Action

From operational mode, run the `show vpls mac-move-action instance` *`instance-name`* command for Device PE1.

```
user@PE1> show vpls mac-move-action instance vpls_1

         Instance: vpls_1
        Local interface: xe-1/3/1.600, Index: 341
     Algorithm used : Base IFL
     Time rec       : 02:29:35
     Recovery timer : Yes
```

Meaning

The output shows Local interface, and Algorithm used indicating that VPLS MAC move has been observed in a VPLS instance.

Verifying the MAC Move Buffer in a VPLS Network

Purpose

Verify the MAC move buffer to monitor the MAC moves that are occurring in the VPLS network.

Action

From operational mode, run the `show l2-learning mac-move-buffer` command for Device PE1.

```
user@PE1> show l2-learning mac-move-buffer

MAC                  Time             Bridge
Address              Rec.             Domain
00:10:00:01:00:09    03:26:00         __vpls_1__
00:10:00:01:00:05    03:26:00         __vpls_1__
00:10:00:01:00:03    03:26:00         __vpls_1__
00:10:00:01:00:05    03:26:00         __vpls_1__
00:10:00:01:00:08    03:26:00         __vpls_1__
00:10:00:01:00:01    03:26:00         __vpls_1__
00:10:00:01:00:03    03:26:00         __vpls_1__
00:10:00:01:00:01    03:26:00         __vpls_1__
00:10:00:01:00:00    03:26:00         __vpls_1__
```

```
00:10:00:01:00:01    03:26:01          __vpls_1__
00:10:00:01:00:06    03:26:01          __vpls_1__
00:10:00:01:00:02    03:26:01          __vpls_1__
00:10:00:01:00:08    03:26:01          __vpls_1__
00:10:00:01:00:00    03:26:01          __vpls_1__
00:10:00:01:00:01    03:26:01          __vpls_1__
00:10:00:01:00:09    03:26:01          __vpls_1__
```

Verifying the VPLS MAC Table for the Base IFL Approach Algorithm

Purpose

Verify the base learning interfaces for the MAC addresses. .

Action

From operational mode, run the `show vpls mac-table extensive` command to obtain extensive information of VPLS MAC table.

```
user@PE1> show vpls mac-table extensive

MAC address: 00:10:00:01:00:00
  Routing instance: vpls_1
   Bridging domain: __vpls_1__, VLAN : NA
   Learning interface: lsi.1049165
   Base learning interface: lsi.1049165
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
   Epoch: 0                            Sequence number: 1
   Learning mask: 0x00000001

MAC address: 00:10:00:01:00:01
  Routing instance: vpls_1
   Bridging domain: __vpls_1__, VLAN : NA
   Learning interface: lsi.1049165
   Base learning interface: lsi.1049165
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
   Epoch: 0                            Sequence number: 1
   Learning mask: 0x00000001

MAC address: 00:10:00:01:00:02
  Routing instance: vpls_1
```

```
     Bridging domain: __vpls_1__, VLAN : NA
     Learning interface: lsi.1049165
     Base learning interface: lsi.1049165
     Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
     Epoch: 0                            Sequence number: 1
     Learning mask: 0x00000001

  MAC address: 00:10:00:01:00:03
    Routing instance: vpls_1
     Bridging domain: __vpls_1__, VLAN : NA
     Learning interface: lsi.1049165
     Base learning interface: lsi.1049165
     Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
     Epoch: 0                            Sequence number: 1
     Learning mask: 0x00000001
```

Meaning

The output of the command `show vpls mac-table extensive` shows the base learning interface of the MAC address.

Verifying That the Interface Is Disabled

Purpose

Verify that the base learning interface of the MAC address is disabled.

Action

From operational mode, run the `show interfaces` *`interface-name`* command for Device PE1.

```
user@PE1> show interfaces xe-1/3/1.600

Logical interface xe-1/3/1.600 (Index 341) (SNMP ifIndex 2864)
 Flags: Up Link-Layer-Down SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.600 ]  Encapsulation: VLAN-VPLS
Input packets : 2234018970
Output packets: 2234728895
Protocol vpls, MTU: 1518
Flags: Is-Primary
```

Meaning

The flag `link-layer-down` in the output indicates that the interface is disabled.

Verifying the VPLS MAC Table for the Statistical Approach Algorithm

Purpose

Verify the VPLS MAC table for the statistical approach algorithm.

Action

From operational mode, run the `show vpls mac-table extensive` command for Device PE1.

```
user@PE1> show vpls mac-table extensive

MAC address: 00:10:00:01:00:00
  Routing instance: vpls_1
   Bridging domain: __vpls_1__, VLAN : NA
   Learning interface: xe-1/3/1.600
   Base learning interface: NULL
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
   Epoch: 3                            Sequence number: 442
   Learning mask: 0x00000002

MAC address: 00:10:00:01:00:01
  Routing instance: vpls_1
   Bridging domain: __vpls_1__, VLAN : NA
   Learning interface: xe-1/3/1.600
   Base learning interface: NULL
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
   Epoch: 3                            Sequence number: 442
   Learning mask: 0x00000003

MAC address: 00:10:00:01:00:02
  Routing instance: vpls_1
   Bridging domain: __vpls_1__, VLAN : NA
   Learning interface: xe-1/3/1.600
   Base learning interface: NULL
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
   Epoch: 3                            Sequence number: 442
```

```
     Learning mask: 0x00000002

  MAC address: 00:10:00:01:00:03
    Routing instance: vpls_1
     Bridging domain: __vpls_1__, VLAN : NA
     Learning interface: xe-1/3/1.600
     Base learning interface: NULL
     Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
     Epoch: 3                            Sequence number: 442
     Learning mask: 0x00000002
```

Meaning

The `Base learning interface` is null which indicates that the statistical approach is in use.

SEE ALSO

virtual-mac

global-mac-move

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	Starting in Junos OS 14.2, you can configure the router to prevent a loop in a VPLS network.

Understanding MAC Pinning

Junos OS supports MAC pinning to prevent loops on the MX series routers. A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops in Layer 2 bridges and in virtual private LAN service (VPLS) networks. To avoid loops, you can enable the MAC pinning feature on an interface. The MAC pinning feature is applicable only when dynamic learning of MAC addresses over interfaces is enabled.

When you enable MAC pinning on an interface in a bridge domain or VPLS domain, MAC addresses learned over that interface cannot be relearned on any other interface in the same bridge domain or

VPLS domain until the MAC address either ages out on the first interface or is cleared from the MAC table. If a packet with the same MAC address arrives at any other interface in the same bridge domain, it is discarded. This, effectively, controls MAC address moves and prevents the creation of loops in Layer 2 bridges and VPLS domains.

NOTE: If the timeout interval for the MAC addresses is not specified by setting the `mac-table-aging-time` statement, the MAC addresses learned over the MAC pinning interface are pinned to the interface until the default timeout period.

You can configure MAC pinning in a bridging environment and in VPLS routing instances. In a bridging environment, you can enable MAC pinning on an access interface and a trunk interface. You can also enable MAC pinning on an access interface or trunk interface of a virtual switch. To avoid loops in the bridging environment, you can use any of the configurations mentioned previously in this topic. To avoid MAC moves and loops, you can use any one of the 16 different MAC pinning configurations.

The MAC pinning feature can also be enabled for provider backbone bridging (PBB) and Ethernet VPN (EVPN) integration, including customer edge (CE) interfaces and EVPN over PBB core in both all-active or single-active mode.

To configure MAC pinning for PBB-EVPN, include the `mac-pinning` statement at the `[edit routing-instances` *`pbbn`* `protocols evpn]`, where `pbbn` is the PBB routing instance over backbone port (B-component). With this configuration, the dynamically learned MAC addresses in the PBB I-component (customer routing instance) bridge domain over CE interfaces, as well as PBB-MPLS core interfaces are pinned. This prevents MAC move on duplicate MAC detection, avoiding loop creation in a network.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
17.2R1	Starting in Junos OS Release 17.2, the MAC pinning feature is enabled on provider backbone bridging (PBB) and Ethernet VPN (EVPN) integration, including customer edge (CE) interfaces and EVPN over PBB core in both all-active or single-active mode.
16.1	Starting in Junos OS Release 16.1, Junos OS supports MAC pinning to prevent loops on the MX series routers.

RELATED DOCUMENTATION

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | 1179

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | 1181

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) for Logical Systems | 1200

Configuring MAC Pinning on VPLS CE Interface for Logical Systems | 1203

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance for Logical Systems | 1205

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance for Logical Systems | 1207

Configuring MAC Pinning on Access Interfaces for Bridge Domains

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in bridge domains, you can configure MAC pinning.

This topic describes how to configure MAC pinning on an access interface in a bridge domain. A logical interface configured to accept untagged packets is called an *access interface* or *access port*. When an access interface receives a tagged or an untagged packet, the interface's VLAN ID is added to the packet. The packet is then forwarded within the bridge domain that is configured with the matching VLAN ID.

To configure MAC pinning on access interfaces in bridge domains:

1. Configure the interface as an access interface and specify the VLAN ID at the `[edit interfaces]` hierarchy level.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port encapsulation ethernet-bridge
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode access
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id vlan-id
```

2. Configure MAC pinning on the interface by including the `mac-pinning` statement at the `[edit bridge-domains]` hierarchy level.

```
[edit bridge-domains]
user@host# set bridge-domain-name bridge-options interface interfacetype-fpc/pic/port mac-pinning
```

3. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        encapsulation ethernet-bridge;
            unit logical-unit-number {
            family bridge {
                interfaces-mode access;
                    vlan-id vlan-id;
                }
            }
        }
    }
user@host# show bridge-domains
    bridge-domains {
        bridge-domain-name  {
            bridge-options {
                    interfacetype-fpc/pic/port {
                        mac-pinning;
                    }
                }
            }
        }
```

RELATED DOCUMENTATION

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | 1177

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | 1179

mac-pinning

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid MAC moves across interfaces in bridge domains, you can configure MAC pinning.

In this topic, we are configuring MAC pinning on a trunk interface in a bridge domain. A logical interface configured to accept any packet tagged with a VLAN ID that matches a VLAN ID specified in the domain is called a *trunk interface* or *trunk port*. When a trunk interface receives a packet tagged with a VLAN ID that matches the list of VLAN IDs specified within the bridge domain, the packet is then forwarded within the bridge domain that is configured with the matching VLAN ID.

To configure MAC pinning on trunk interfaces in bridge domains:

1. Configure the interface as a trunk interface and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the bridge domain by specifying the name of the bridge and the VLAN ID.

```
[edit bridge-domains]
user@host# set bridge-domain-name vlan-id all
```

3. Configure MAC pinning on the interface by including the `mac-pinning` statement at the [`edit switch-options`] hierarchy level.

```
[edit switch-options]
user@host# set interface interfacetype-fpc/pic/port mac-pinning
```

4. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number {
            family bridge {
                interfaces-mode trunk ;
                vlan-id-list vlan-id-numbers;
                }
            }
        }
        }
    user@host# show bridge-domains
    bridge-domains {
        bridge-domain-name {
            vlan-id all;
            }
        }
user@host# show switch-options
    switch-options {
    interface interfacetype-fpc/pic/port {
    mac-pinning;
}
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

mac-pinning

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual switches, you can configure MAC pinning.

This topic describes how to configure MAC pinning on an access interface for a bridge domain in a virtual switch. A virtual switch represents a Layer 2 network as it filters and forwards traffic only at the data link layer. Each bridge domain in a virtual switch participates in Layer 2 Learning and Forwarding. When an access interface for a bridge domain receives a tagged or untagged packet, the interface's VLAN ID is added to the packet. The packet is then forwarded within the bridge domain that is configured with the matching VLAN ID.

To configure MAC pinning on access interfaces for bridge domains in a virtual switch:

1. Configure the interface as an access interface and specify the VLAN ID.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port encapsulation ethernet-bridge
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode access
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id vlan-id
```

2. Configure the routing instance as a virtual switch and specify the access interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type virtual-switch
user@host# set routing-instance-name interface interfacetype-fpc/pic/port
```

3. Configure the bridge domain and specify the list of VLAN IDs.

```
[edit routing-instances]
user@host# set routing-instance-name bridge-domains bridge-domain-name vlan-id-list vlan-id-numbers
```

4. Configure MAC pinning on the interface.

```
[edit routing-instances]
user@host# set routing-instance-name bridge-domains bridge-domain-name bridge-options interface
interfacetype-fpc/pic/port mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        encapsulation ethernet-bridge;
            unit logical-unit-number {
            family bridge {
                interface-mode access;
                    vlan-id vlan-id;
                }
            }
        }
    }
user@host# show routing-instances
    routing-instances{
```

```
    routing-instance-name {
    instance-type virtual-switch;
        interface interfacetype-fpc/pic/port;
        }
    }
user@host# show bridge-domains
    bridge-domains {
        bridge-domain-name {
            vlan-id -list vlan-id-numbers;
            bridge-options {
                    interface interfacetype-fpc/pic/port {
                        mac-pinning;
                    }
                }
            }
        }
```

RELATED DOCUMENTATION

Configuring MAC Pinning on Access Interfaces for Bridge Domains | 1176

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | 1181

mac-pinning

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual switches, you can configure MAC pinning.

This topic describes how to configure MAC pinning on a trunk interface for a bridge domain in a virtual switch. A virtual switch represents a Layer 2 network as it filters and forwards traffic only at the data link layer. Each bridge domain in a virtual switch participates in Layer 2 Learning and Forwarding. When a trunk interface for a bridge domain receives a packet tagged with a VLAN ID that matches the list of VLAN IDs specified with the bridge domain, the packet is then forwarded within the bridge domain that is configured with the matching VLAN ID.

To configure MAC pinning on trunk interfaces for bridge domains in a virtual switch:

1. Configure the interface as a trunk interface and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the routing instance as virtual switch and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type virtual-switch;
user@host# set routing-instance-name interface interfacetype-fpc/pic/port
```

3. Configure the bridge domain and specify the list of VLAN IDs.

```
[edit routing-instances]
user@host# set routing-instance-name bridge-domains bridge-domain-name vlan-id-list vlan-id-numbers
```

4. Configure MAC pinning on the interface by including the `mac-pinning` statement at the [`edit switch-options`] hierarchy level.

```
[edit switch-options]
user@host# set interface interfacetype-fpc/pic/port mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number {
            family bridge {
                interface-mode trunk ;
                vlan-id-list vlan-id-numbers;
                }
            }
        }
    }
user@host# show routing-instances
    routing-instances{
    routing-instance-name{
    instance-type virtual-switch;
        interface interfacetype-fpc/pic/port;
        }
    }
user@host# show bridge-domains
    bridge-domains {
        bridge-domain-name{
            vlan-id -list  vlan-id-numbers;
            }
            }
            }
        }
        user@host# show switch-options
        switch-options {
        interface interfacetype-fpc/pic/port {
        mac-pinning;
    }
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

mac-pinning

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP)

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning on a trunk interface for all pseudowires of the VPLS routing instance.

To configure MAC pinning for the VPLS routing instance:

1. Configure the interface as a trunk interface and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number encapsulation vlan-vpls;
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the routing instance as VPLS and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type vpls;
user@host# set routing-instance-name interface interfacetype-fpc/pic/port
```

3. Configure MAC pinning after specifying the routing protocol configuration.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls mac-pinning
```

4. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number {
            encapsulation vlan-vpls;
            }
            family bridge {
                interface-mode trunk ;
                vlan-id-list vlan-id-numbers;
                }
            }
        }
     user@host# show routing-instances
    routing-instances{
```

```
    routing-instance-name {
        instance-type vpls;
        interface interfacetype-fpc/pic/port;
    protocols{
        vpls {
        mac-pinning;
        }
    }
}
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

mac-pinning

Configuring MAC Pinning on VPLS CE Interface

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning on a VPLS customer edge (CE) trunk interface.

To configure MAC pinning on a VPLS CE interface:

1. Configure the trunk interface and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number encapsulation vlan-vpls;
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the VPLS routing instance and specify the CE interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type vpls;
user@host# set routing-instance-name interface interfacetype-fpc/pic/port
```

3. Configure MAC pinning on the VPLS CE interface.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interfacetype-fpc/pic/port mac-pinning
```

4. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number {
            encapsulation vlan-vpls;
            }
            family bridge {
                interface-mode trunk ;
                vlan-id-list  vlan-id-numbers;
                }
            }
        }
    }
    user@host# show routing-instances
    routing-instances{
    routing-instance-name{
        instance-type vpls;
        interface interfacetype-fpc/pic/port;
    protocols{
        vpls {
            interface interfacetype-fpc/pic/port {
        mac-pinning;
        }
    }
}
}
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

mac-pinning

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning for all pseudowires of the VPLS site in a BGP-based VPLS routing instance.

To configure MAC pinning for all pseudowires in a BGP-based VPLS Routing instance:

1. Configure the interface as a trunk interface and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number encapsulation vlan-vpls;
user@host# set interfacetype-fpc/pic/port unit logical-unit-number  family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number  family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the VPLS routing instance and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type vpls;
user@host# set routing-instance-name interface interfacetype-fpc/pic/port
```

3. Configure MAC pinning after specifying the routing protocol and the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interface interfacetype-fpc/pic/port
user@host# set routing-instance-name protocols vpls site site-name mac-pinning
```

4. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number  {
            encapsulation vlan-vpls;
            family bridge {
                interface-mode trunk ;
                vlan-id-list vlan-id-numbers;
                }
            }
        }
    }
    user@host# show routing-instances
        routing-instances{
    routing-instance-name  {
        instance-type vpls;
        interface interfacetype-fpc/pic/port;
    protocols{
        vpls {
        interface interfacetype-fpc/pic/port;
            site site-name  {
                mac-pinning;
            }
        }
    }
}
}
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

mac-pinning

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning on all pseudowires of a specific neighbor of an LDP-based VPLS routing instance.

To configure MAC pinning on all pseudowires of a specific neighbor in a VPLS routing instance with LDP signaling:

1. Configure the interfaces as trunk interfaces and specify the list of VLAN IDs.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
user@host# set interfacetype-fpc/pic/port unit logical-unit-number encapsulation vlan-vpls;
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge interface-mode trunk
user@host# set interfacetype-fpc/pic/port unit logical-unit-number family bridge vlan-id-list vlan-id-
numbers
```

2. Configure the VPLS routing instance and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type vpls;
user@host# set routing-instance-name interface interfacetype-fpc/pic/port;
```

3. Configure the routing protocol and specify the trunk interface and the VPLS identifier.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interface interfacetype-fpc/pic/port
user@host# set routing-instance-name protocols vpls vpls-id vpls-id
```

4. Configure MAC pinning after specifying the neighbor ID.

```
user@host# set routing-instance-name protocols vpls neighbor neighbor-id mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
    interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit logical-unit-number {
            encapsulation vlan-vpls;
            family bridge {
                interface-mode trunk ;
                vlan-id-list vlan-id-numbers;
                }
            }
        }
    }
    user@host# show routing-instances
        routing-instances{
    routing-instance-name  {
        instance-type vpls;
        interface interfacetype-fpc/pic/port;
    protocols{
        vpls {
            interface interfacetype-fpc/pic/port;
        vpls-id vpls-id;
            neighbor neighbor-id  {
                mac-pinning;
            }
        }
    }
}
}
```

RELATED DOCUMENTATION

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance | **1187**

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

mac-pinning

Configuring MAC Pinning on Access Interfaces for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in logical systems, you can configure MAC pinning. A set of logical systems within a single router can handle the functions previously handled by several small routers.

This topic describes how to configure MAC pinning on access interfaces of a logical system.

To configure MAC pinning on access interfaces in logical systems:

1. Configure the access interface.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port encapsulation ethernet-vpls
```

2. Create the logical system by specifying the name of the logical system, and specify the access interface and list of VLAN IDs associated with the interface.

```
[edit]
user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
unit-number family bridge interface-mode access vlan-id vlan-id
```

3. Configure MAC pinning on the access interface by including the [`mac-pinning`] statement in the [`edit bridge-domains`] hierarchy.

```
[edit bridge-domains]
user@host# set bridge-domain-name bridge-options interface interfacetype-fpc/pic/port mac-pinning
```

4. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        encapsulation ethernet-vpls;
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    family bridge {
                        interface-mode access;
                        vlan-id vlan-id;
                    }
                }
            }
        }
```

```
user@host# show bridge-domains
bridge-domains {
            bridge-domain-name  {
                bridge-options {
                    interface interfacetype-fpc/pic/port {
                        mac-pinning;
                    }
                }
            }
        }
```

```
    }
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | 1174

Configuring MAC Pinning on Access Interfaces for Bridge Domains | 1176

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | 1179

mac-pinning

Configuring MAC Pinning on Trunk Interfaces for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in logical systems, you can configure MAC pinning. A set of logical systems within a single router can handle the functions previously handled by several small routers.

This topic describes how to configure MAC pinning on trunk interfaces of a logical system.

To configure MAC pinning on trunk interfaces in logical systems:

1. Configure the interface and specify the encapsulation.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
```

2. Create the logical system by specifying the name of the logical system, and specify the trunk interface and list of VLAN IDs associated with the interface.

```
[edit]
user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
unit-number encapsulation vlan-vpls family bridge interface-mode trunk vlan-id-list vlan-id-numbers
```

3. Configure the bridge domain by specifying the VLAN ID parameter.

```
[edit bridge-domains]
user@host# set bridge-domain-name vlan-id all
```

4. Configure MAC pinning on the trunk interface by including the [mac-pinning] statement at the [edit switch-options] hierarchy level.

```
[edit switch-options]
user@host# set interface interfacetype-fpc/pic/port mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name  {
        interfaces {
            interfacetype-fpc/pic/port
{
                  unit 0 {
                    encapsulation vlan-vpls;
                    family bridge {
                        interface-mode trunk;
                        vlan-id-list vlan-id-numbers;
                    }
                }
            }
        }
```

```
user@host# show bridge-domains
bridge-domains {
            bridge-domain-name  {
                vlan-id all;
            }
        }
```

```
user@host# show switch-options
switch-options {
            interface interfacetype-fpc/pic/port {
                mac-pinning;
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

Configuring MAC Pinning on Access Interfaces for Logical Systems | **1191**

mac-pinning

Configuring MAC Pinning on Access Interfaces in Virtual Switches for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual switches in logical systems, you can configure MAC pinning. A set of logical systems within a single router can handle the functions previously handled by several small routers.

This topic describes how to configure MAC pinning on access interfaces in virtual switch routing instance of a logical system.

To configure MAC pinning on access interfaces in virtual switches for logical systems:

1. Configure the interface and specify the encapsulation.

   ```
   [edit interfaces]
   user@host# set interfacetype-fpc/pic/port encapsulation ethernet-vpls
   ```

2. Create the logical system by specifying the name of the logical system, and specify the access interface.

   ```
   [edit ]
   user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
   unit-number family bridge interface-mode access vlan-id vlan-id
   ```

3. Configure the routing instance as virtual switch and specify the access interface.

   ```
   [edit routing-instances]
   user@host# set routing-instance-name instance-type virtual-switch;
   user@host# set routing-instance-name interface interfacetype-fpc/pic/port
   ```

4. Configure the bridge domain and specify the list of VLAN IDs.

   ```
   [edit routing-instances]
   user@host# set routing-instance-name bridge-domains bridge-domain-name vlan-id-list vlan-id-numbers
   user@host# set routing-instance-name bridge-domains bridge-domain-name bridge-options interface
   interfacetype-fpc/pic/port mac-pinning
   ```

5. In configuration mode, verify the configuration.

   ```
   user@host# show interfaces
   interfaces {
       interfacetype-fpc/pic/port {
           encapsulation ethernet-vpls;
   ```

```
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    family bridge {
                        interface-mode access;
                        vlan-id vlan-id;
                    }
                }
            }
        }
```

```
user@host# show routing-instances
routing-instances {
            routing-instance-name {
                instance-type virtual-switch;
                interface interfacetype-fpc/pic/port;
                bridge-domains {
                    bridge-domain-name  {
                        vlan-id-list vlan-id-numbers;
                        bridge-options {
                            interface interfacetype-fpc/pic/port {
                                mac-pinning;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | 1174

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | 1179

mac-pinning

Configuring MAC Pinning on Trunk Interfaces in Virtual Switches for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual switches in logical systems, you can configure MAC pinning. A set of logical systems within a single router can handle the functions previously handled by several small routers.

This topic describes how to configure MAC pinning on trunk interfaces in virtual switch routing instance of a logical system.

To configure MAC pinning on trunk interfaces in virtual switches for logical systems:

1. Configure the interface and specify the encapsulation.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
```

2. Create the logical system by specifying the name of the logical system and specify the trunk interface.

```
[edit]
user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
unit-number family bridge interface-mode trunk vlan-id-list vlan-id-numbers
```

3. Configure the routing instance as a virtual switch and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name  instance-type virtual-switch;
user@host# set routing-instance-name  interface interfacetype-fpc/pic/port
```

4. Configure the bridge domain and specify the list of VLAN IDs.

```
[edit routing-instances]
user@host# set routing-instance-name  bridge-domains bridge-domain-name  vlan-id-list vlan-id-numbers
```

5. Configure MAC pinning on the trunk interface at the `[edit switch-options]` hierarchy level.

```
[edit switch-options]
user@host# set interface interfacetype-fpc/pic/port mac-pinning
```

6. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
    }
}
```

```
user@host# show logical-systems
logical-systems {
logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                 unit logical-unit-number {
                   encapsulation vlan-vpls;
                   family bridge {
                       interface-mode trunk;
                       vlan-id-list vlan-id-numbers;
                   }
               }
            }
        }
```

```
user@host# show routing-instances
routing-instances {
            routing-instance-name {
                instance-type virtual-switch;
                interface interfacetype-fpc/pic/port;
                bridge-domains {
                    bridge-domain-name  {
                        vlan-id-list vlan-id-numbers;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
user@host# show switch-options
switch-options {
            interface interfacetype-fpc/pic/port {
                mac-pinning;
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on Access Interfaces in Virtual Switches for Logical Systems | **1195**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

mac-pinning

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual switches in logical systems, you can configure MAC pinning. A set of logical systems within a single router can handle the functions previously handled by several small routers.

This topic describes how to configure MAC pinning for all pseudowires of the VPLS routing instance for logical systems.

To configure MAC pinning for all pseudowires of the VPLS routing instance for logical systems:

1. Configure the interface by specifying the encapsulation.

   ```
   [edit interfaces]
   user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
   user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
   ```

2. Create the logical system by specifying the name and the trunk interface on the logical unit.

   ```
   [edit]
   user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
   unit-number  encapsulation vlan-vpls family bridge interface-mode trunk vlan-id-list vlan-id-numbers
   ```

3. Configure the VPLS routing instance.

   ```
   [edit routing-instances]
   user@host# set routing-instance-name  instance-type vpls
   user@host# set routing-instance-name  interface interfacetype-fpc/pic/port
   user@host# set routing-instance-name  protocols vpls mac-pinning
   ```

4. In configuration mode, verify the configuration.

   ```
   user@host# show interfaces
   interfaces {
       interfacetype-fpc/pic/port {
           flexible-vlan-tagging;
           encapsulation flexible-ethernet-services;
       }
   }
   ```

   ```
   user@host# show logical-systems
   logical-systems {
       logical-system-name {
   ```

```
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    encapsulation vlan-vpls;
                    family bridge {
                        interface-mode trunk;
                        vlan-id-list vlan-id-numbers;
                    }
                }
            }
        }
```

```
user@host# show routing-instances
routing-instances {
            routing-instance-name {
                instance-type vpls;
                interface interfacetype-fpc/pic/port;
                protocols {
                    vpls {
                        mac-pinning;
                    }
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning for All Pseudowires of the VPLS Routing Instance (LDP and BGP) | **1183**

mac-pinning

Configuring MAC Pinning on VPLS CE Interface for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning on a VPLS customer edge (CE) trunk interface in a logical system.

To configure MAC pinning for VPLS customer edge interface for logical systems:

1. Configure the interfaces by specifying the encapsulation.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
```

2. Create the logical system by specifying the name and the trunk interface on the logical unit.

```
[edit]
user@host# set logical-systems logical-system-nameinterfaces interfacetype-fpc/pic/port unit logical-
unit-number  encapsulation vlan-vpls family bridge interface-mode trunk vlan-id-list vlan-id-numbers
```

3. Configure the VPLS CE interface and VPLS routing instance.

```
[edit routing-instances]
user@host# set routing-instance-name  instance-type vpls;
user@host# set routing-instance-name  interface interfacetype-fpc/pic/port
```

4. Configure MAC pinning on the VPLS CE interface.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interfacetype-fpc/pic/port mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    encapsulation vlan-vpls;
                    family bridge {
                        interface-mode trunk;
                        vlan-id-list vlan-id-numbers;
                    }
                }
            }
        }
```

```
user@host# show routing-instances
routing-instances{
routing-instance-name  {
    instance-type vpls;
    interface interfacetype-fpc/pic/port;
protocols{
    vpls {
        interface interfacetype-fpc/pic/port {
    mac-pinning;
    }
}
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on VPLS CE Interface | **1185**

mac-pinning

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning for all pseudowires of the VPLS site in a BGP-based VPLS routing instance for a logical system.

To configure MAC pinning for all pseudowires in a BGP-based VPLS Routing instance for a logical system:

1. Configure the interface by specifying the encapsulation.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
```

2. Create the logical system and specify the trunk interface and list of VLAN IDs on the logical unit.

```
[edit]
user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
unit-number encapsulation vlan-vpls family bridge interface-mode trunk vlan-id-list vlan-id-numbers
```

3. Configure the VPLS routing instance and specify the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name  instance-type vpls;
user@host# set routing-instance-name  interface interfacetype-fpc/pic/port
```

4. Configure MAC pinning after specifying the routing protocol and the trunk interface.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interface interfacetype-fpc/pic/port
user@host# set routing-instance-name protocols vpls site site-name mac-pinning
```

5. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    encapsulation vlan-vpls;
                    family bridge {
                        interface-mode trunk;
                        vlan-id-list vlan-id-numbers;
                    }
                }
            }
        }
```

```
routing-instances{
routing-instance-name  {
    instance-type vpls;
    interface interfacetype-fpc/pic/port;
protocols{
    vpls {
    interface interfacetype-fpc/pic/port;
        site site-name  {
            mac-pinning;
        }
    }
}
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | 1174

Configuring MAC Pinning for All Pseudowires of the VPLS Site in a BGP-Based VPLS Routing Instance | 1187

mac-pinning

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance for Logical Systems

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. To avoid loops across interfaces in virtual private LAN services (VPLS), you can configure MAC pinning.

This topic describes how to configure MAC pinning on all pseudowires of a specific neighbor of an LDP-based VPLS routing instance for logical systems.

To configure MAC pinning on all pseudowires of a specific neighbor in a VPLS routing instance with LDP signaling for logical systems:

1. Configure the interfaces by specifying the encapsulation.

```
[edit interfaces]
user@host# set interfacetype-fpc/pic/port flexible-vlan-tagging
user@host# set interfacetype-fpc/pic/port encapsulation flexible-ethernet-services
```

2. Create the logical system interface by specifying the name of the logical system and specifying the trunk interface on the logical unit.

```
[edit]
user@host# set logical-systems logical-system-name interfaces interfacetype-fpc/pic/port unit logical-
unit-number encapsulation vlan-vpls family bridge interface-mode trunk vlan-id-list vlan-id-numbers
```

3. Configure the VPLS routing instance and specify the interface.

```
[edit routing-instances]
user@host# set routing-instance-name  instance-type vpls
user@host# set routing-instance-name  interface interfacetype-fpc/pic/port
```

4. Configure the routing protocol and specify the trunk interface and the VPLS identifier.

```
[edit routing-instances]
user@host# set routing-instance-name protocols vpls interface interfacetype-fpc/pic/port
user@host# set routing-instance-name protocols vpls vpls-id vpls-id
```

5. Configure MAC pinning after specifying the neighbor ID.

```
user@host# set routing-instance-name protocols vpls neighbor neighbor-id mac-pinning
```

6. In configuration mode, verify the configuration.

```
user@host# show interfaces
interfaces {
    interfacetype-fpc/pic/port {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
    }
}
```

```
user@host# show logical-systems
logical-systems {
    logical-system-name {
        interfaces {
            interfacetype-fpc/pic/port {
                unit logical-unit-number  {
                    encapsulation vlan-vpls;
                    family bridge {
                        interface-mode trunk;
                        vlan-id-list vlan-id-numbers;
                    }
                }
            }
        }
```

```
user@host# show routing-instance
routing-instances{
routing-instance-name  {
    instance-type vpls;
    interface interfacetype-fpc/pic/port;
protocols{
    vpls {
        interface interfacetype-fpc/pic/port;
    vpls-id vpls-id;
        neighbor neighbor-id {
            mac-pinning;
        }
    }
}
}
```

RELATED DOCUMENTATION

Understanding MAC Pinning | 1174

Configuring MAC Pinning on All Pseudowires of a Specific Neighbor of LDP-Based VPLS Routing Instance | 1189

mac-pinning

Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Access Interfaces

IN THIS SECTION

- Requirements | 1210
- Overview | 1210
- Configuration | 1211
- Verification | 1214

This example shows how to avoid loops in bridge domains by enabling the MAC pinning feature on access interfaces.

Requirements

This example uses the following hardware and software components:

- MX Series 5G Universal Routing Platforms
- Junos OS Release 16.1 running on the routers

Overview

IN THIS SECTION

● Topology | **1210**

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. Loops can occur in Layer 2 bridges and in VPLS networks. To avoid loops, you can enable the MAC pinning feature on the interfaces. The MAC pinning feature is applicable only when dynamic learning of MAC addresses over interfaces is enabled.

This example shows how to enable MAC pinning on two access interfaces in a bridge domain.

Topology

In this example, you configure the interfaces `ge-4/0/6` and `xe-4/2/0` on the MX Series router as access interfaces. Access interfaces accept both untagged and tagged packets and forward the packets within a specified bridge domain, `bd1`. Specify **1** as the VLAN ID for the interfaces and the bridge domain. When an untagged or a tagged packet is received on any of the access interfaces, the packet is accepted, the VLAN ID is added to the packet, and the packet is forwarded within the bridge domain that is configured with the matching ID.

In the bridge domain, after specifying the VLAN ID, specify **131071** as the maximum number of MAC addresses that can be learned on the access interfaces and specify **1048575** as the size of the MAC address table for the bridge domain or VLAN.

In this topology, frequent MAC moves can occur, which can result in loops. To prevent these loops, you can configure MAC pinning. When you configure MAC pinning on an interface, the MAC address learned on the interface cannot be learned on another interface in the same bridge domain. For example, configure MAC pinning on the access interface `ge-4/0/6.` When a packet is received on this interface, the packet is accepted, the VLAN ID is added and the packet is forwarded within the bridge domain with the matching ID. However, if a packet with the same MAC address is received on any other access interface, say `xe-4/2/0,` the packet is discarded or dropped as that MAC address is pinned to the access interface `ge-4/0/6` . This behavior is common to all access interfaces configured on the router, regardless of whether access pinning is enabled on the access interface or not.

Configuration

IN THIS SECTION

- CLI Quick Configuration | 1211
- Procedure | 1211
- Results | 1213

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 interfaces ge-4/0/6 encapsulation ethernet-bridge
set interfaces ge-4/0/6 unit 0 family bridge interface-mode access
set interfaces ge-4/0/6 unit 0 family bridge vlan-id 1
set interfaces xe-4/2/0 encapsulation ethernet-bridge
set interfaces xe-4/2/0 unit 0 family bridge interface-mode access
set interfaces xe-4/2/0 unit 0 family bridge vlan-id 1
set bridge-domains bd1 vlan-id 1
set bridge-domains bd1 bridge-options mac-table-size 1048575
set bridge-domains bd1 bridge-options interface ge-4/0/6.0 interface-mac-limit 131071
set bridge-domains bd1 bridge-options interface ge-4/0/6.0 mac-pinning
set bridge-domains bd1 bridge-options interface xe-4/2/0.0 interface-mac-limit 131071
set bridge-domains bd1 bridge-options interface xe-4/2/0.0 mac-pinning
```

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*

To configure MAC pinning on access interfaces in bridge domains:

1. Configure both the interfaces as access interfaces and specify the VLAN ID.

```
[edit interfaces]
user@host# set interfaces ge-4/0/6 encapsulation ethernet-bridge
user@host# set interfaces ge-4/0/6 unit 0 family bridge interface-mode access
user@host# set interfaces ge-4/0/6 unit 0 family bridge vlan-id 1
user@host# set interfaces xe-4/2/0 encapsulation ethernet-bridge
user@host# set interfaces xe-4/2/0 unit 0 family bridge interface-mode access
user@host# set interfaces xe-4/2/0 unit 0 family bridge vlan-id 1
```

2. Specify the name of the bridge domain.

```
[edit bridge-domains]
user@host# set bridge-domains bd1 vlan-id 1
```

3. Specify the size of the MAC address table for the bridge domain.

```
[edit bridge-domains]
user@host#  set bridge-domains bd1 bridge-options mac-table-size 1048575
```

4. Specify the maximum number of MAC addresses that can be learned on both the access interfaces.

```
[edit bridge-domains]
user@host# set bridge-domains bd1 bridge-options interface ge-4/0/6.0 interface-mac-limit 131071
user@host#  set bridge-domains bd1 bridge-options interface xe-4/2/0.0 interface-mac-limit 131071
```

5. Configure MAC pinning on both the access interfaces.

```
[edit bridge-domains]
user@host#  set bridge-domains bd1 bridge-options interface ge-4/0/6.0 mac-pinning
user@host#  set bridge-domains bd1 bridge-options interface xe-4/2/0.0 mac-pinning
```

Results

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

```
user@host# show interfaces
ge-4/0/6 {
encapsulation ethernet-bridge;
    unit 0 {
        family bridge {
            interface-mode access;
        vlan-id 1;
        }
    }
}
xe-4/2/0 {
encapsulation ethernet-bridge;
    unit 0 {
        family bridge {
            interface-mode access;
        vlan-id 1;
        }
    }
}

user@host# show bridge-domains
bridge-domains {
    bd1 {
        vlan-id 1;
        bridge-options {
            mac-table-size {
                1048575;
                }
                interface ge-4/0/6.0 {
                interface-mac-limit {
                131071;
                }
                mac-pinning;
            }
            interface xe-4/2/0.0 {
                interface-mac-limit {
                131071;
                }
                mac-pinning;
            }
        }
    }
}
```

If you have completed configuring the device, enter `commit` from the configuration mode.

Verification

IN THIS SECTION

- Verifying That MAC Pinning Is Configured Correctly | 1214

Verifying That MAC Pinning Is Configured Correctly

Purpose

Ensure that MAC pinning has been enabled on the access interfaces.

Action

From operational mode, enter the `show l2-learning interface` command.

```
user@host> show l2-learning interface
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
xe-4/0/6.0                     131000                               MP
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
Routing Instance Name : default-switch
```

```
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
xe-4/2/0.0                     131000                               MP
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
```

Meaning

The `Interface flags` field indicates the interfaces that have MAC pinning enabled.

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on Access Interfaces for Bridge Domains | **1176**

Configuring MAC Pinning on Access Interfaces for Bridge Domains in a Virtual Switch | **1179**

mac-pinning

Example: Prevention of Loops in Bridge Domains by Enabling the MAC Pinnning Feature on Trunk Interfaces

IN THIS SECTION

- Requirements | **1216**
- Overview | **1216**
- Configuration | **1217**
- Verification | **1223**

This example shows how to avoid loops in bridge domains by enabling the MAC pinning feature on trunk interfaces.

Requirements

This example uses the following hardware and software components:

- MX Series 5G Universal Routing Platforms
- Junos OS Release 16.1 running on the routers

Overview

IN THIS SECTION

● Topology | **1216**

A MAC move occurs when a MAC address frequently appears on a different physical interface than the one it was learned on. Frequent MAC moves indicate the presence of loops. Loops can occur in Layer 2 bridges and in VPLS networks. To avoid loops, you can enable the MAC pinning feature on the interfaces. The MAC pinning feature is applicable only when dynamic learning of MAC addresses over interfaces is enabled.

This example shows how to enable MAC pinning on three aggregated trunk interfaces in a bridge domain.

Topology

In this example, you configure the interfaces `xe-0/1/1` and `xe-0/3/1` on the MX Series router as an aggregated Ethernet interface, `ae1`. You can configure the other four interfaces, `ge-4/0/6`, `ge-4/1/6`, `xe-4/2/0`, and `xe-4/3/0`, as aggregated Ethernet interfaces, `ae2` and `ae3`. Each of these aggregated Ethernet interfaces are configured as trunk interfaces. Trunk interfaces accept only tagged packets and forward the packets within a specified bridge domain, `BD_Trunk_all`. Specify the list of VLAN IDs for the interfaces and the bridge domain. When a tagged packet is received on any of the aggregated trunk interfaces, the packet is accepted, and the packet is forwarded within the bridge domain that is configured with the matching ID.

In the bridge domain, after specifying the VLAN ID, specify **131000** as the maximum number of MAC addresses that can be learned on each of the aggregated Ethernet trunk interfaces and specify **1048575** as the size of the MAC address table for the bridge domain or VLAN.

In this topology, frequent MAC moves can occur, which can result in loops. To prevent these loops, you can configure MAC pinning. When you configure MAC pinning on an interface, the MAC address learned on the interface cannot be learned on another interface in the same bridge domain. For example, configure MAC pinning on the aggregated Ethernet interface `ae1`. When a packet is received on this interface, the packet is accepted, and the packet is forwarded within the bridge domain with the matching ID. However, if a packet with the same MAC address is received on any other trunk interface, say `ae2`, the packet is discarded or dropped as that MAC address is pinned to the trunk interface `ae1`. This behavior is common to all trunk interfaces configured on the router, regardless of whether access pinning is enabled on the trunk interface or not.

Configuration

IN THIS SECTION

- CLI Quick Configuration | **1217**
- Procedure | **1218**
- Results | **1220**

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 interfaces xe-0/1/1 gigether-options 802.3ad ae1
set interfaces xe-0/3/1 gigether-options 802.3ad ae1
set interfaces ge-4/0/6 gigether-options 802.3ad ae3
set interfaces ge-4/1/6 gigether-options 802.3ad ae3
set interfaces xe-4/2/0 gigether-options 802.3ad ae2
set interfaces xe-4/3/0 gigether-options 802.3ad ae2
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation flexible-ethernet-services
set interfaces ae1 unit 0 family bridge interface-mode trunk
set interfaces ae1 unit 0 family bridge vlan-id-list 1-5
set interfaces ae2 flexible-vlan-tagging
set interfaces ae2 encapsulation flexible-ethernet-services
set interfaces ae2 unit 0 family bridge interface-mode trunk
set interfaces ae2 unit 0 family bridge vlan-id-list 1-5
set interfaces ae3 flexible-vlan-tagging
set interfaces ae3 encapsulation flexible-ethernet-services
set interfaces ae3 unit 0 family bridge interface-mode trunk
set interfaces ae3 unit 0 family bridge vlan-id-list 1-5
set bridge-domains BD_Trunk_all vlan-id-list 1-5
set bridge-domains BD_Trunk_all bridge-options mac-table-size 1048575
set bridge-domains BD_Trunk_all bridge-options interface ae1.0 interface-mac-limit 131000
set bridge-domains BD_Trunk_all bridge-options interface ae2.0 interface-mac-limit 131000
set bridge-domains BD_Trunk_all bridge-options interface ae3.0 interface-mac-limit 131000
set switch-options interface ae1.0 interface-mac-limit 131000
set switch-options interface ae1.0 mac-pinning
set switch-options interface ae2.0 interface-mac-limit 131000
set switch-options interface ae2.0 mac-pinning
set switch-options interface ae3.0 interface-mac-limit 131000
set switch-options interface ae3.0 mac-pinning
```

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*

To configure MAC pinning on trunk interfaces in bridge domains:

1. Configure the interfaces as member aggregated Ethernet interfaces.

   ```
   [edit interfaces]
   user@host# set interfaces xe-0/1/1 gigether-options 802.3ad ae1
   user@host# set interfaces xe-0/3/1 gigether-options 802.3ad ae1
   user@host# set interfaces ge-4/0/6 gigether-options 802.3ad ae3
   user@host# set interfaces ge-4/1/6 gigether-options 802.3ad ae3
   user@host# set interfaces xe-4/2/0 gigether-options 802.3ad ae2
   user@host# set interfaces xe-4/3/0 gigether-options 802.3ad ae2
   ```

2. Configure the aggregated Ethernet interfaces as trunk interfaces and specify the list of VLAN IDs.

   ```
   user@host# set interfaces ae1 flexible-vlan-tagging
   user@host# set interfaces ae1 encapsulation flexible-ethernet-services
   user@host# set interfaces ae1 unit 0 family bridge interface-mode trunk
   user@host# set interfaces ae1 unit 0 family bridge vlan-id-list 1-5
   user@host# set interfaces ae2 flexible-vlan-tagging
   ```

```
user@host# set interfaces ae2 encapsulation flexible-ethernet-services
user@host# set interfaces ae2 unit 0 family bridge interface-mode trunk
user@host# set interfaces ae2 unit 0 family bridge vlan-id-list 1-5
user@host# set interfaces ae3 flexible-vlan-tagging
user@host# set interfaces ae3 encapsulation flexible-ethernet-services
user@host# set interfaces ae3 unit 0 family bridge interface-mode trunk
user@host# set interfaces ae3 unit 0 family bridge vlan-id-list 1-5
```

3. Specify the name of the bridge domain.

```
[edit bridge-domains]
user@host# set bridge-domains BD_Trunk_all vlan-id-list 1-5
```

4. Specify the size of the MAC address table for the bridge domain.

```
user@host#  set bridge-domains BD_Trunk_all bridge-options mac-table-size 1048575
```

5. Specify the maximum number of MAC addresses that can be learned on all three trunk interfaces.

```
user@host# set bridge-domains BD_Trunk_all bridge-options interface ae1.0 interface-mac-limit 131000
user@host#  set bridge-domains BD_Trunk_all bridge-options interface ae2.0 interface-mac-limit 131000
user@host#  set bridge-domains BD_Trunk_all bridge-options interface ae3.0 interface-mac-limit 131000
```

6. Configure MAC pinning on each of the aggregated Ethernet interfaces at the `[edit switch-options]` hierarchy level.

```
[edit switch-options]
user@host# set switch-options interface ae1.0 interface-mac-limit 131000
user@host# set switch-options interface ae1.0 mac-pinning
user@host# set switch-options interface ae2.0 interface-mac-limit 131000
user@host# set switch-options interface ae2.0 mac-pinning
user@host# set switch-options interface ae3.0 interface-mac-limit 131000
user@host# set switch-options interface ae3.0 mac-pinning
```

Results

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

```
user@host# show interfaces
interfaces {
    xe-0/0/0 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 1-5;
            }
        }
    }
    xe-0/1/1 {
        gigether-options {
            802.3ad ae1;
        }
    }
    xe-0/3/1 {
        gigether-options {
            802.3ad ae1;
        }
    }
    ge-4/0/6 {
        gigether-options {
            802.3ad ae3;
        }
    }
    ge-4/1/6 {
        gigether-options {
            802.3ad ae3;
        }
    }
    xe-4/2/0 {
        gigether-options {
            802.3ad ae2;
        }
    }
    xe-4/3/0 {
        gigether-options {
            802.3ad ae2;
        }
    }
    ae1 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 1-5;
            }
        }
    }
    ae2 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 1-5;
            }
        }
    }
    ae3 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list 1-5;
            }
        }
    }
}

user@host# show bridge-domains
bridge-domains {
    BD_Trunk_all {
        vlan-id-list 1-5;
        bridge-options {
            mac-table-size {
                1048575;
            }
            interface ae1.0 {
                interface-mac-limit {
                    131000;
                }
            }
            interface ae2.0 {
                interface-mac-limit {
                    131000;
                }
            }
            interface ae3.0 {
                interface-mac-limit {
                    131000;
                }
            }
        }
    }
}

user@host# show switch-options
switch-options {
    interface ae1.0 {
        interface-mac-limit {
            131000;
        }
        mac-pinning;
    }
    interface ae2.0 {
        interface-mac-limit {
            131000;
        }
        mac-pinning;
    }
    interface ae3.0 {
        interface-mac-limit {
            131000;
        }
        mac-pinning;
    }
}
```

If you have completed configuring the device, enter `commit` from the configuration mode.

Verification

IN THIS SECTION

- Verifying that MAC Pinning Is Configured Correctly | 1223

Verifying that MAC Pinning Is Configured Correctly

Purpose

Ensure that MAC pinning is enabled on the trunk interfaces.

Action

From operational mode, enter the `show l2-learning interface` command.

```
user@host> show l2-learning interface
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
xe-0/0/0.0                     8192
                    BD_Tru..   1024       Forwarding
                    BD_Tru..   1024       Forwarding
                    BD_Tru..   1024       Forwarding
                    BD_Tru..   1024       Forwarding
                    BD_Tru..   1024       Forwarding
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ae1.0                          131000                               MP
```

```
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ae2.0                          131000                               MP
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down, MP - MAC Pinning)
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ae3.0                          131000                               MP
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
                    BD_Tru..   131000     Forwarding
```

Meaning

The `Interface flags` field indicates the interfaces that have MAC pinning enabled.

RELATED DOCUMENTATION

Understanding MAC Pinning | **1174**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains | **1177**

Configuring MAC Pinning on Trunk Interfaces for Bridge Domains in a Virtual Switch | **1181**

mac-pinning

Configuring Improved VPLS MAC Address Learning on T4000 Routers with Type 5 FPCs

Junos OS Release 12.3 enables improved virtual private LAN service (VPLS) MAC address learning on T4000 routers with Type 5 FPCs by supporting up to 262,143 MAC addresses per VPLS routing instance. In Junos OS releases before Release 12.3, T4000 routers with Type 5 FPCs support only 65,535 MAC addresses per VPLS routing instance.

Before you begin, configure VPLS. See "Configuring VPLS Routing Instances" on page 675.

To enable improved VPLS MAC address learning on T4000 routers with Type 5 FPCs:

1. Enable the network services mode by including the `enhanced-mode` statement at the `[edit chassis network-services]` hierarchy level.

   ```
   [edit chassis network-services]
   user@host# set enhanced-mode
   ```

   NOTE: After you configure the `enhanced-mode` statement and commit your configuration, a warning message prompts you to reboot the router.

2. Perform a system reboot in operational mode.

   ```
   user@host> request system-reboot
   ```

   After the router reboots, only the T4000 Type 5 FPCs are online while the remaining FPCs are offline. You can verify the status of the FPCs by using the `show chassis fpc` operational mode command.

3. Modify the size of the VPLS MAC address table at the `[edit routing-instance` *`instance-name`* `protocols vpls]` hierarchy level.

   ```
   [edit routing-instance instance-name  protocols vpls]
   user@host# set mac-table-size size
   ```

For example, to set the MAC address learning limit to 262,143 addresses for each `vpls` routing instance:

```
[edit routing-instance vpls protocols vpls]
user@host# set mac-table-size 262143
```

NOTE: The `enhanced-mode` statement supports up to 262,143 MAC addresses per VPLS routing instance. However, the MAC address learning limit for each interface remains the same (that is, 65,535 MAC addresses).

4. In configuration mode, verify the configuration.

```
user@host# show routing-instances instance-name
vpls {
    instance-type vpls;
    protocols {
        vpls {
            mac-table-size {
                262143;
            }
        }
    }
}
```

To disable the improved VPLS MAC address learning feature on T4000 routers with Type 5 FPCs, include the `delete chassis network-services enhanced-mode` statement at the `[edit]` hierarchy level.

NOTE: After you disable network services mode and commit your configuration, a warning message prompts you to reboot the router. You must reboot the router. Continuing without a reboot might result in unexpected system behavior.

RELATED DOCUMENTATION

enhanced-mode

Network Services Mode Overview

Configuring VPLS Routing Instances | 675

show chassis fpc

Understanding Qualified MAC Learning

IN THIS SECTION

● Qualified MAC Learning on the First, Second, and Third VLAN Tags | 1227

MAC learning is the process by which a device learns the MAC addresses of all the nodes on a network.

When a node is first connected to an Ethernet LAN or VLAN, it has no information about the other nodes on the network. As data is sent through the network, data packets include a data frame listing their source and destination MAC addresses. The data frame is forwarded to a target port, which is connected to the second device. The MAC address is learned locally at the target port, which facilitates communications for frames that later enter the target port and contain addresses previously learned from a received frame.

During MAC learning, on a ingress packet, the outer tag is implicitly removed (using the `pop` operation) and the learning happens on the inner tag. MAC learning is preceded by VLAN manipulation. VLAN used for learning can be changed by VLAN push/pop/swap operations.

Qualified MAC learning enables a device to learn the MAC addresses of network nodes by determining the innermost VLAN tag of single-tagged, 2-tagged , or 3-tagged ingress packets without deleting the outer tag (using the `pop` operation). If the ingress packet has one tag, learning happens on VLAN 4096, and no tags are implicitly removed. If the ingress packet has two tags, MAC learning happens on the second VLAN and no tags are implicitly removed. If the ingress packet has more than three tags, all tags beyond the third tag are treated as part of data and are not considered for MAC learning.

Qualified MAC Learning on the First, Second, and Third VLAN Tags

For a single-tagged ingress packet, qualified MAC learning happens on VLAN 4096, which is the default VLAN.

In the case of a 2-tagged ingress packet, you enable qualified MAC learning on the second (inner) tag by using the `vlan-id inner-all` configuration statement on the VPLS routing instance. Learning on the second tag happens without the implicit removal of the first (outer) tag. If the ingress packet has more than two tags, all tags beyond the second tag are treated as part of data and are not considered for learning.

Similarly, for an 3-tagged ingress packet, you enable qualified MAC learning on the third (innermost) tag by configuring the `deep-vlan-qualified-learning` *`vlan_tag_number`* statement on the logical interface along with the `vlan-id inner-all` statement on the routing instance. Qualified MAC learning happens on the third tag, and no VLAN manipulation happens on the outer tags. However, if `deep-vlan-qualified-learning` *vlan_tag_number* is enabled to learn on the third VLAN and the ingress packet has only two VLANs, the qualified MAC learning happens on the default VLAN 4096.

Note the following points while configuring qualified MAC learning:

- A logical interface contained in a VPLS routing instance configured with `vlan-id inner-all` might or might not have `deep-vlan-qualified-learning` *vlan_tag_number* configured.
- A logical interface configured with `deep-vlan-qualified-learning` *vlan_tag_number*, must belong to a VPLS routing instance that also has `vlan-id inner-all` configured.
- A logical interface configured with `deep-vlan-qualified-learning` *vlan_tag_number*, must also be configured with one outer and one inner tag.

RELATED DOCUMENTATION

deep-vlan-qualified-learning

vlan-id inner-all

Qualified Learning VPLS Routing Instance Behavior

The following tables summarize the VPLS routing instance behavior regarding qualified MAC learning.

Table 25 on page 1229 summarizes the scenario when `vlan-id all` is not configured on a routing instance. All VLAN identifiers specified on the logical interfaces are included in the routing instance.

Table 25: VPLS Routing Instance Behavior When vlan-id all Is Not Configured

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map (no VLAN-id on the routing-instance)	Outer and inner tag only	Outer tag and inner range tag with VLAN-map	Outer tag and inner range	LSI
No tag	Drop (only works with native VLAN-id)	Drop (only works with native VLAN-id)	Drop	Drop	Drop	Drop	Learning on VLAN 4096
1 tag	OK	OK	Drop	Drop	Drop	Drop	Learning on outer tag
1 tag mismatch	Drop	Drop	Drop	Drop	Drop	Drop	Learning on outer tag

Table 25: VPLS Routing Instance Behavior When vlan-id all Is Not Configured *(Continued)*

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map (no VLAN-id on the routing-instance)	Outer and inner tag only	Outer tag and inner range tag with VLAN-map	Outer tag and inner range	LSI
2 tags	OK	OK	OK (swap/ pop-swap: MAC learned on vpls db but no VLAN-id)	OK (MAC learned on VPLS database if no VLAN-id on routing instance; MAC learning on routing instance VLAN-id otherwise)	OK (MAC learning on outer VLAN after VLAN map)	OK (MAC learning on outer VLAN)	Learning on outer tag
2 tags mismatch	Drop	Drop	Drop	Drop	Drop	Drop	Learning on outer tag
> 2 tags	OK	OK	Same as with 2 tags	Same as with 2 tags	Same as with 2 tags	Same as with 2 tags	Learning on outer tag

Table 26 on page 1231 summarizes the scenario when `vlan-id all` is configured on a routing instance. All VLAN identifiers specified on the logical interfaces are included in the routing instance.

Table 26: VPLS Routing Instance Behavior when vlan-id all is configured

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map	Outer and inner tag	Outer tag and inner range tag with VLAN-map	Outer tag and inner range	LSI
No tag	Drop	Drop	Invalid configuration	Drop	Invalid configuration	Drop	Learning on VLAN 4096
1 tag	OK, learning on the tag	OK, learning on the tag	Invalid configuration	Drop	Invalid configuration	Drop	Learning on the tag
1 tag mismatch	Drop	Drop	Invalid configuration	Drop	Invalid configuration	Drop	Learning on the tag
2 tags	OK, learning on the tag	OK, learning on the tag	Invalid configuration	Pop outer tag, MAC learning on inner tag	Invalid configuration	Pop, MAC learning on inner tag	Learning on the outer tag
2 tags mismatch	Drop	Drop	Invalid configuration	Drop	Invalid configuration	Drop	Learning on the outer tag

Table 26: VPLS Routing Instance Behavior when vlan-id all is configured *(Continued)*

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map	Outer and inner tag	Outer tag and inner range tag with VLAN-map	Outer tag and inner range	LSI
> 2 tags	OK, learning on the tag	OK, learning on the tag	Invalid configuration	Pop outer tag, learning on the second tag	Invalid configuration	Pop, learning on the second tag	Learning on the outer tag

Table 27 on page 1232 summarizes the scenario when `vlan-id inner-all` is configured on a routing instance, but `deep-vlan-qualified-learning` *`vlan_tag_number`* is not configured on the logical interface .

Table 27: VPLS Routing Instance and Logical Interface Behavior When vlan-id inner-all Is Configured

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map	Outer and inner tags only	Outer tag and inner range tag with VLAN-map	Outer tag and inner tag range	LSI
No tag	Drop (OK with native VLAN-id)	Drop	Drop	Drop	Drop	Drop	Learning on VLAN 4096
1 tag	OK (MAC learning on VLAN 4096)	OK (MAC learning on VLAN 4096)	Drop	Drop	Drop	Drop	Learning on VLAN 4096

Table 27: VPLS Routing Instance and Logical Interface Behavior When vlan-id inner-all Is Configured *(Continued)*

Configured tags on Logical Interface							
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map	Outer and inner tags only	Outer tag and inner range tag with VLAN-map	Outer tag and inner tag range	LSI
1 tag mismatch	Drop	Drop	Drop	Drop	Drop	Drop	Learning on VLAN 4096
2 tags	OK (MAC learning on the inner VLAN)	OK (MAC learning on the inner VLAN)	OK (MAC learning on inner VLAN after VLAN-map operation	OK (MAC learning on the inner VLAN). No implicit popping of outer tag.	OK (MAC learning on the inner VLAN after VLAN-map operation).	OK (MAC learning on inner VLAN). No implicit popping of outer tag.	Learning on the inner tag
2 tags mismatch	Drop	Drop	Drop	Drop	Drop	Drop	Learning on the inner tag
> 2 tags	OK (MAC learning on the 2nd VLAN tag)	OK (MAC learning on the 2nd VLAN tag)	OK (MAC learning on the 2nd VLAN after VLAN-map operation).	OK (MAC learning on the 2nd incoming VLAN). No popping of outer tag.	OK (MAC learning on the 2nd VLAN after VLAN-map operation)	OK (MAC learning on the 2nd incoming VLAN). No popping of outer tag.	Learning on the 2nd tag

Table 28 on page 1234 summarizes the scenario when `vlan-id inner-all` is configured on a routing instance and `deep-vlan-qualified-learning` *3* is configured on the logical interface .

Table 28: VPLS Routing Instance and Logical Interface Behavior When vlan-id inner-all and deep-vlan-qualified-learning Is Configured

Configured tags on Logical Interface						
Incoming tags	Outer tag only	Outer tag range only	Outer and inner tags with VLAN-map	Outer and inner tags only	Outer tag and inner range tag with VLAN-map	Outer tag and inner tag range
No tag	Invalid configuration	Invalid configuration	Drop	Invalid configuration	Invalid configuration	Invalid configuration
1 tag	Invalid configuration	Invalid configuration	Drop	Invalid configuration	Invalid configuration	Invalid configuration
1 tag mismatch	Invalid configuration	Invalid configuration	Drop	Invalid configuration	Invalid configuration	Invalid configuration
2 tags	Invalid configuration	Invalid configuration	OK (learning on VLAN 4096)	OK (learning on VLAN 4096)	Invalid configuration	Invalid configuration
2 tags mismatch	Invalid configuration	Invalid configuration	Drop	Invalid configuration	Invalid configuration	Invalid configuration
3 tags	Invalid configuration	Invalid configuration	OK (MAC learning on innermost VLAN)	OK (MAC learning on innermost VLAN)	Invalid configuration	Invalid configuration

RELATED DOCUMENTATION

Understanding Qualified MAC Learning | 1227

Configuring Qualified MAC Learning

Qualified MAC learning enables a device to learn the MAC addresses of network nodes by learning the innermost VLAN tag of single vlan-tagged, 2-tagged , or 3-tagged ingress packets without deleting the outer tag (using the `pop` operation).

To enable qualified MAC learning on the innermost VLAN tag for 2 or 3 tagged packets:

1. In configuration mode, go to the `[edit routing-instances]` hierarchy level.

   ```
   [edit]
   user@host# edit routing-instances instance-name
   ```

2. Include the `vlan-id inner-all` statement.

   ```
   [edit interfaces instance-name]
   user@host# set vlan vlan-id inner-all
   ```

For enabling qualified MAC learning on the third VLAN tag (innermost) of a 3-tagged packet, along with `vlan-id inner-all`, configure the `deep-vlan-qualified-learning` *`vlan_tag_number`* statement.

1. In configuration mode, go to the `edit interfaces` *`interface-name`* `unit` *`logical_unit_number`* hierarchy level.

   ```
   [edit]
   user@host# edit interfaces interface-name unit logical_unit_number
   ```

2. Include the `deep-vlan-qualified-learning` *`vlan_tag_number`* statement.

   For bidirectional traffic flow, include the statement `input-vlan-map pop`.

   ```
   [edit interfaces interface--name unit logical_unit_number]
   user@host# set deep-vlan-qualified-learning 3
   user@host# set input-vlan-map pop
   ```

3. Verify learned MAC address information..

   ```
   user@host# show vpls mac-table
   ```

RELATED DOCUMENTATION

Understanding Qualified MAC Learning | **1227**

deep-vlan-qualified-learning

vlan-id inner-all

CHAPTER 40

Configuring Class of Service and Firewall Filters in VPLS

IN THIS CHAPTER

- Configuring EXP-Based Traffic Classification for VPLS | **1237**
- Configuring Firewall Filters and Policers for VPLS | **1238**
- Firewall Filter Match Conditions for VPLS Traffic | **1243**

Configuring EXP-Based Traffic Classification for VPLS

You can enable EXP classification on traffic entering core facing VPLS LSI interfaces on a VPLS routing instance by configuring either a logical tunnel interface (`lt-`) or the `no-tunnel-services` statement. By configuring either of these, a default EXP classifier is enabled on every core facing interface that includes **family mpls** in its configuration. This feature works on MX Series routers and EX Series switches only. You can configure an EXP classifier explicitly at the `[edit class-of-service]` hierarchy level. For more information about EXP classifiers, see the Class of Service User Guide (Routers and EX9200 Switches).

To enable EXP classification on traffic entering core facing VPLS LSI interfaces on a VPLS routing instance, include the `no-tunnel-services` statement:

```
no-tunnel-services;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances` *`routing-instance-name`* `protocols vpls]`
- `[edit logical-systems` *`logical-system-name`* `routing-instances` *`routing-instance-name`* `protocols vpls]`

RELATED DOCUMENTATION

Configuring Firewall Filters and Policers for VPLS | **1238**

Firewall Filter Match Conditions for VPLS Traffic | 1243

Configuring Firewall Filters and Policers for VPLS

IN THIS SECTION

- Configuring a VPLS Filter | 1239
- Configuring a VPLS Policer | 1242

You can configure both firewall filters and policers for VPLS. Firewall filters allow you to filter packets based on their components and to perform an action on packets that match the filter. Policers allow you to limit the amount of traffic that passes into or out of an interface.

VPLS filters and policers act on a Layer 2 frame that includes the media access control (MAC) header (after any VLAN rewrite or other rules are applied), but does not include the cyclical redundancy check (CRC) field.

You can apply VPLS filters and policers on the PE router to customer-facing interfaces only.

NOTE: In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

NOTE: The behavior of firewall filters processing with MAC addresses differs between DPCs and MPCs. On MPCs, interface filters are always applied before MAC learning occurs. The input forwarding table filter is applied after MAC learning is completed. However, on DPCs, MAC learning occurs independently of the application of filters. If the CE-facing interface of the PE where the firewall filter is applied is an MPC, then the MAC entry times out and is never learned again. However, if the CE-facing interface of the PE where the firewall filter is applied is an DP, then the MAC entry is not timed out and if the MAC address entry is manually cleared, it is relearned.

The following sections explain how to configure filters and policers for VPLS:

Configuring a VPLS Filter

To configure a filter for VPLS, include the `filter` statement at the `[edit firewall family vpls]` hierarchy level:

```
[edit firewall family vpls]
filter filter-name {
    interface-specific;
    term term-name {
        from {
            match-conditions;
        }
        then {
            actions;
        }
    }
}
```

For more information about how to configure firewall filters, see the Routing Policies, Firewall Filters, and Traffic Policers User Guide. For information on how to configure a VPLS filter match condition, see "Firewall Filter Match Conditions for VPLS Traffic" on page 1243.

To configure a filter for VPLS traffic, complete the following tasks:

Configuring an Interface-Specific Counter for VPLS

When you configure a firewall filter for VPLS and apply it to multiple interfaces, you can specify individual counters specific to each interface. This allows you to collect separate statistics on the traffic transiting each interface.

To generate an interface-specific counter for VPLS, you configure the `interface-specific` statement. A separate instantiation of the filter is generated. This filter instance has a different name (based on the interface name) and collects statistics on the interface specified only.

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

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

NOTE: The counter name is restricted to 24 bytes. If the renamed counter exceeds this maximum length, it might be rejected.

Configuring an Action for the VPLS Filter

You can configure the following actions for a VPLS filter at the `[edit firewall family vpls filter filter-name term term-name then]` hierarchy level: `accept`, `count`, `discard`, `forwarding-class`, `loss-priority`, `next`, `policer`.

Configuring VPLS FTFs

Forwarding table filters (FTFs) are filters configured for forwarding tables. For VPLS, they are attached to the destination MAC (DMAC) forwarding table of the VPLS routing instance. You define VPLS FTFs in the same manner as any other type of FTF. You can only apply a VPLS FTF as an input filter.

To specify a VPLS FTF, include the `filter input` statement at the `[edit routing-instance routing-instance-name forwarding-options family vpls]` hierarchy level:

```
[edit routing-instance routing-instance-name forwarding-options family vpls]
filter input filter-name;
```

Changing Precedence for Spanning-Tree BPDU Packets

Spanning tree BPDU packets are automatically set to a high precedence. The queue number on these packets is set to 3. A queue value of 3 indicates high precedence. To enable this higher precedence on BPDU packets, an instance-specific BPDU precedence filter named `default_bpdu_filter` is automatically attached to the VPLS DMAC table. This filter places a high precedence on all packets sent to `01:80:c2:00:00:00/24`.

You can overwrite this filter by configuring a VPLS FTF filter and applying it to the VPLS routing instance. For more information, see "Configuring VPLS FTFs" on page 1240 and "Applying a VPLS Filter to a VPLS Routing Instance" on page 1241.

Applying a VPLS Filter to an Interface

To apply a VPLS filter to an interface, include the `filter` statement:

```
filter {
    group index;
    input input-filter-name;
    output output-filter-name;
}
```

You can include this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit number family vpls]`
- `[edit logical-systems logical-system-name interfaces interface-name unit number family vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

In the `input` statement, list the name of the VPLS filter to be evaluated when packets are received on the interface. In the `output` statement, list the name of the VPLS filter to be evaluated when packets are transmitted on the interface.

NOTE: For output interface filters, MAC addresses are learned after the filter action is completed. When an output interface filter's action is `discard`, the packet is dropped before the MAC address is learned. However, an input interface filter learns the MAC address before discarding the packet.

Applying a VPLS Filter to a VPLS Routing Instance

You can apply a VPLS filter to a VPLS routing instance. The filter checks traffic passing through the specified routing instance.

Input routing instance filters learn the MAC address before the filter action is completed, so if the filter action is `discard`, the MAC address is learned before the packet is dropped.

To apply a VPLS filter to packets arriving at a VPLS routing instance and specify the filter, include the `filter input` statement at the `[edit routing-instances routing-instance-name forwarding-options family vpls]` hierarchy level:

```
[edit routing-instances routing-instance-name forwarding-options family vpls]
filter input input-filter-name;
```

Configuring a Filter for Flooded Traffic

You can configure a VPLS filter to filter flooded packets. CE routers typically flood the following types of packets to PE routers in VPLS routing instances:

- Layer 2 broadcast packets

- Layer 2 multicast packets
- Layer 2 unicast packets with an unknown destination MAC address
- Layer 2 packets with a MAC entry in the DMAC routing table

You can configure filters to manage how these flooded packets are distributed to the other PE routers in the VPLS routing instance.

To apply a flooding filter to packets arriving at the PE router in the VPLS routing instance, and specify the filter, include the `flood input` statement:

```
flood input filter-name;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-instances routing-instance-name forwarding-options family vpls]`
- `[edit logical-systems logical-system-name routing-instances routing-instance-name forwarding-options family vpls]`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

Configuring a VPLS Policer

You can configure a policer for VPLS traffic. The VPLS policer configuration is similar to the configuration of any other type of policer.

VPLS policers have the following characteristics:

- You cannot police the default VPLS routes stored in the flood table from PE router–sourced flood traffic.
- When specifying policing bandwidth, the VPLS policer considers all Layer 2 bytes in a packet to determine the packet length.

To configure a VPLS policer, include the `policer` statement at the `[edit firewall]` hierarchy level:

```
[edit firewall]
policer policer-name {
    bandwidth-limit limit;
    burst-size-limit limit;
    then action;
}
```

To apply a VPLS policer to an interface, include the `policer` statement:

```
policer {
    input input-policer-name;
    output output-policer-name;
}
```

You can include this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit number family vpls]`
- `[edit logical-systems logical-system-name interfaces interface-name unit number family vpls`

NOTE: ACX Series routers do not support the `[edit logical-systems]` hierarchy.

In the `input` statement, list the name of the VPLS policer to be evaluated when packets are received on the interface. In the `output` statement, list the name of the VPLS policer to be evaluated when packets are transmitted on the interface. This type of VPLS policer can only apply to unicast packets. For information about how to filter flood packets, see "Configuring a Filter for Flooded Traffic" on page 1241.

RELATED DOCUMENTATION

Routing Policies, Firewall Filters, and Traffic Policers User Guide

Firewall Filter Match Conditions for VPLS Traffic | 1243

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 29 on page 1244 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. .

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 29: 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 29: Firewall Filter Match Conditions for VPLS 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. 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 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*	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`	Do not match destination prefixes in the specified list. For more information, see the `destination-prefix-list` match condition.

Table 29: Firewall Filter Match Conditions for VPLS 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 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`*	Do not match on the DSCP. For details, see the `dscp` match condition.

Table 29: 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`	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 29: 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 29: 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 29: Firewall Filter Match Conditions for VPLS Traffic *(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 `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`*	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 29: 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.
`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.
`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`*	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`*	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.
`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).

Table 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ip-precedence-except` *`ip-precedence-field`*	Do not match on the IP precedence field.
`ip-protocol` *`number`*	IP protocol field.
`ip-protocol-except` *`number`*	Do not match on the IP protocol field.
`ip-source-address` *`address`*	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*	Match the IPv6 source address in a *named-list*.
`ipv6-address` *address*	128-bit address that supports the standard syntax for IPv6 addresses.
`ipv6-destination-address` *address*	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.
`ipv6-destination-prefix-list` *named-list*	Match the IPv6 destination addresses in a *named-list*.

Table 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-next-header` *protocol*	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 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
	• **vrrp**—Virtual Router Redundancy Protocol
`ipv6-next-header-except` *protocol*	Do not match the IPv6 next header protocol type.

Table 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`ipv6-payload-protocol` *protocol*	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 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
	• **vrrp**—Virtual Router Redundancy Protocol
`ipv6-payload-protocol-except` *protocol*	Do not match the IPv6 payload protocol.
`ipv6-prefix-list` *named-list*	Match the IPv6 address in a *named-list*.
`ipv6-source-address` *address*	128-bit address that is the originating source node address for this packet.
`ipv6-traffic-class` *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)

Table 29: 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*	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
`learn-vlan-1p-priority-except` *number*	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
`learn-vlan-dei`	Match the user VLAN ID drop eligability indicator (DEI) bit.
`learn-vlan-dei-except`	Do not match the user VLAN ID 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 29: 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`. For IP traffic 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`*	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`*	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.
`prefix-list` *`name`*	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.

Table 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`prefix-list` *name* `except`	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*	TCP or UDP source port field. You cannot specify the `port` and `source-port` match conditions in the same term.
`source-port-except` *number*	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*	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`	Do not match the source prefixes in the specified prefix list. For more information, see the `source-prefix-list` match condition.

Table 29: 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`*	Specify `broadcast`, `multicast`, `unknown-unicast`, or `known-unicast`.
`traffic-type-except` *`type-name`*	Do not match on the traffic type. Specify `broadcast`, `multicast`, `unknown-unicast`, or `known-unicast`.

Table 29: Firewall Filter Match Conditions for VPLS Traffic *(Continued)*

Match Condition	Description
`user-vlan-1p-priority` *`number`*	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.
`user-vlan-1p-priority-except` *`number`*	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.
`user-vlan-id` *`number`*	Match the first VLAN identifier that is part of the payload.
`user-vlan-id-except` *`number`*	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: For matches `flexible-match-mask` and `flexible-match-range` match-start layer-4 used to match over IPV6 header will not work for L2 family filters such as "bridge, CCC, VPLS". Instead, use layer-3 with appropriate offset to match over IPV6 payload fields.

NOTE: Commit check issues an error if `traffic-type` *`known-unicast`* or `traffic-type` *`unknown-unicast`* is unsupported.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

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

CHAPTER 41

Monitoring and Tracing VPLS

IN THIS CHAPTER

- Configuring Port Mirroring for VPLS Traffic | **1263**
- Configuring Y.1731 Functionality for VPLS to Support Delay and Delay Variation | **1263**
- Tracing VPLS Traffic and Operations | **1265**

Configuring Port Mirroring for VPLS Traffic

Use Feature Explorer for platform support and release information for all features, including Port Mirroring for VPLS Traffic.

To configure port mirroring for VPLS include the `port-mirroring` statement at the `[edit forwarding-options]` hierarchy level. For more information about configuring port mirroring for VPLS for all platforms supported, see Routing Policies, Firewall Filters, and Traffic Policers User Guide.

Configuring Y.1731 Functionality for VPLS to Support Delay and Delay Variation

For VPLS, you can configure the Ethernet frame delay measurement (ETH-DM) functionality to trigger two-way ETH-DM and allow concurrent ETH-DM CLI sessions from the same local maintenance association end point (MEP). The feature also provides the option to perform ETH-DM for a given 802.1q priority, to set the size of the data type, length, and value (TLV), to disable the `session-id-tlv` option, and to generate XML output.

Use Feature Explorer to determine if a feature is supported on your platform and release.

This feature complements the ITU-T Y.1731 Ethernet service OAM feature. On-demand delay measurement for VPLS is supported on MX Series routers installed with Rev-B DPCs. Only the two-way delay measurement feature is supported for VPLS connections.

MX Series routers with modular port concentrators (MPCs) and 10-Gigabit Ethernet MPCs with SFP+ support ITU-T Y.1731 functionality on VPLS for frame-delay and delay-variation.

This feature is currently supported only for up MEPs. Set the MEP direction to up by configuring the `up` option for the `direction` statement at the `[edit protocols oam ethernet connectivity-fault-management maintenance-domain` *`md-name`* `maintenance-association` *`ma-name`* `mep` *`mep-id`*`]` hierarchy level.

This feature also provides support for an optional configuration where you can delegate the server-side processing (for two-way delay measurement) to the Packet Forwarding Engine (PFE) to prevent overloading on the Routing Engine. To enable this feature, include the `delegate-server-processing` statement at the `[edit protocols oam ethernet connectivity-fault-management performance-monitoring]` hierarchy level. By default, the server-side processing is done by the Routing Engine.

The following commands enable you to monitor and maintain the Y.1731 feature for VPLS:

- To display the delay measurement values across a VPLS connection, use the `monitor ethernet delay-measurement two-way (`*`remote-mac-address`* `| mep` *`mep-id`*`) maintenance-domain` *`name`* `maintenance-association` *`name`* `count` *`count`* `wait` *`time`* `priority 802.1p-value` *`size`* `no-session-id-tlv xml` command.

- The feature also provides support for enhanced continuity measurement by using an existing continuity check protocol. The continuity for every remote MEP is measured as the percentage of time that a remote MEP was operationally up over the total administratively enabled time.

  To display the continuity measurement information, use the `show oam ethernet connectivity-fault-management mep-database maintenance-domain` *`name`* `maintenance-association` *`name`* `local-mep` *`identifier`* `remote-mep` *`identifier`* command.

- You can restart the continuity measurement by clearing the currently measured operational uptime and administrative enabled time. To clear the existing continuity measurement and restart counting the operational uptime, use the `clear oam ethernet connectivity-fault-management continuity-measurement maintenance-domain` *`name`* `maintenance-association` *`name`* `local-mep` *`identifier`* `remote-mep` *`identifier`* command.

- To clear the delay statistics, issue a `clear oam ethernet connectivity-fault-management statistics` command or a `clear oam ethernet connectivity-fault-management delay-statistics two-way maintenance-domain` *`md-name`* `maintenance-association` *`ma-name`* command.

RELATED DOCUMENTATION

Ethernet Interfaces User Guide for Routing Devices

Configuring MEP Interfaces to Support Ethernet Frame Delay Measurements

Example: Configuring Two-Way Ethernet Frame Delay Measurements with Single-Tagged Interfaces

Tracing VPLS Traffic and Operations

To trace VPLS traffic, include the `traceoptions` statement:

```
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier> <disable>;
}
```

You can include this statement at the following hierarchy levels:

- `[edit logical-systems logical-system-name routing-instances routing-instance-name protocols vpls]`
- `[edit routing-instances routing-instance-name protocols vpls]`

The following trace flags display the operations associated with VPLS:

- `all`—All VPLS tracing options
- `connections`—VPLS connections (events and state changes)
- `error`—Error conditions
- `nlri`—VPLS advertisements received or sent using BGP
- `route`—Trace-routing information
- `topology`—VPLS topology changes caused by reconsideration or advertisements received from other PE routers using BGP

8

PART

Connecting Layer 2 VPNs and Circuits to Other VPNs

Connecting Layer 2 VPNs to Other VPNs | **1267**

Connecting Layer 2 Circuits to Other VPNs | **1333**

CHAPTER 42

Connecting Layer 2 VPNs to Other VPNs

IN THIS CHAPTER

- Layer 2 VPN to Layer 2 VPN Connections | **1267**
- Using the Layer 2 Interworking Interface to Interconnect a Layer 2 VPN to a Layer 2 VPN | **1268**
- Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN | **1270**
- Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview | **1292**
- Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN | **1294**
- Configure an Anycast Gateway in an L2VPN | **1325**
- Stitching Layer 2 VPNs and VPWS | **1328**

Layer 2 VPN to Layer 2 VPN Connections

As the need to link different Layer 2 services to one another for expanded service offerings grows, Layer 2 MPLS VPN services are increasingly in demand. Junos OS enables you to terminate Layer 2 VPN into Layer 2 VPN (also known as Layer 2 VPN stitching) using the Layer 2 interworking (iw0) interface.

Another way to do this is to use a Tunnel Services PIC to loop packets out and back from the Packet Forwarding Engine (PFE), to link together Layer 2 networks. The Layer 2 interworking software interface avoids the need for the Tunnel Services PIC and overcomes the limitation of bandwidth constraints imposed by the Tunnel Services PIC.

RELATED DOCUMENTATION

Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN | **1270**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 VPN to a Layer 2 VPN

Instead of using a physical Tunnel PIC for looping the packet received from the Layer 2 VPN to another Layer 2 VPN, the Layer 2 Interworking interface uses Junos OS to stitch together both Layer 2 VPN routes.

To configure the interworking interface, include the `iw0` statement. The `iw0` statement is configured at the `[edit interfaces]` hierarchy level.

```
[edit interfaces]
iw0 {
    unit 0 {
        peer 1;
    }
    unit 1 {
        peer 0;
    }
}
```

NOTE: You must always create and delete (or deactivate) interworking (iw) interfaces in pairs, such as:

```
set interfaces iw0 unit 0 peer-unit 1
set interfaces iw0 unit 1 peer-unit 0
```

An error message displays if you delete or deactivate only one of the interworking interfaces. To successfully deactivate the interfaces and avoid any configuration errors, you must deactivate both.

The configuration of an interworking (iw) interface is similar to the configuration of a logical tunnel (lt) interface. In this example, the logical interfaces must be associated with the endpoints of both Layer 2 VPN connections terminating on this router. To make the association, include the `interfaces` statement and specify `iw0` as the interface name. Include the statement at the `[edit routing-instances` *`routing-instances-name`* `protocols l2vpn site` *`site-name`*`]` hierarchy level for each routing instance. The `routing-instances` statement is configured at the `[edit routing-instances]` hierarchy level.

```
[edit routing-instances]
L2VPN-PE1 {
    instance-type l2vpn;
    interface iw0.0;
    route-distinguisher 65000:3;
    vrf-target target:65000:2;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site CE3 {
                site-identifier 3;
                interface iw0.0 {
                    remote-site-id 1;
                }
            }
        }
    }
}
L2VPN-PE5 {
    instance-type l2vpn;
    interface iw0.1;
    route-distinguisher 65000:33;
    vrf-target target:65000:2;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site CE3 {
                site-identifier 3;
                interface iw0.1 {
                    remote-site-id 5;
                }
            }
        }
    }
}
```

In addition to the `iw0` interface configuration, Layer 2 interworking `l2iw` protocols need to be configured. Without the `l2iw` configuration, the `l2iw` routes will not be formed, regardless of whether any `iw` interfaces are present. Only standard trace options can be configured within the l2iw protocol. The minimum configuration necessary for the feature to work is shown below:

```
[edit]
protocols {
    l2iw;
}
```

RELATED DOCUMENTATION

Layer 2 VPN Applications | 145

Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN | 1270

Example: Interconnecting a Layer 2 VPN with a Layer 2 VPN

IN THIS SECTION

- Requirements | 1270
- Overview and Topology | 1271
- Configuration | 1272

This example provides a step-by-step procedure for interconnecting and verifying a Layer 2 VPN with a Layer 2 VPN. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- 2 MX Series 5G Universal Routing Platforms
- 2 M Series Multiservice Edge Routers
- 1 T Series Core Routers
- 1 EX Series Ethernet Switches

Overview and Topology

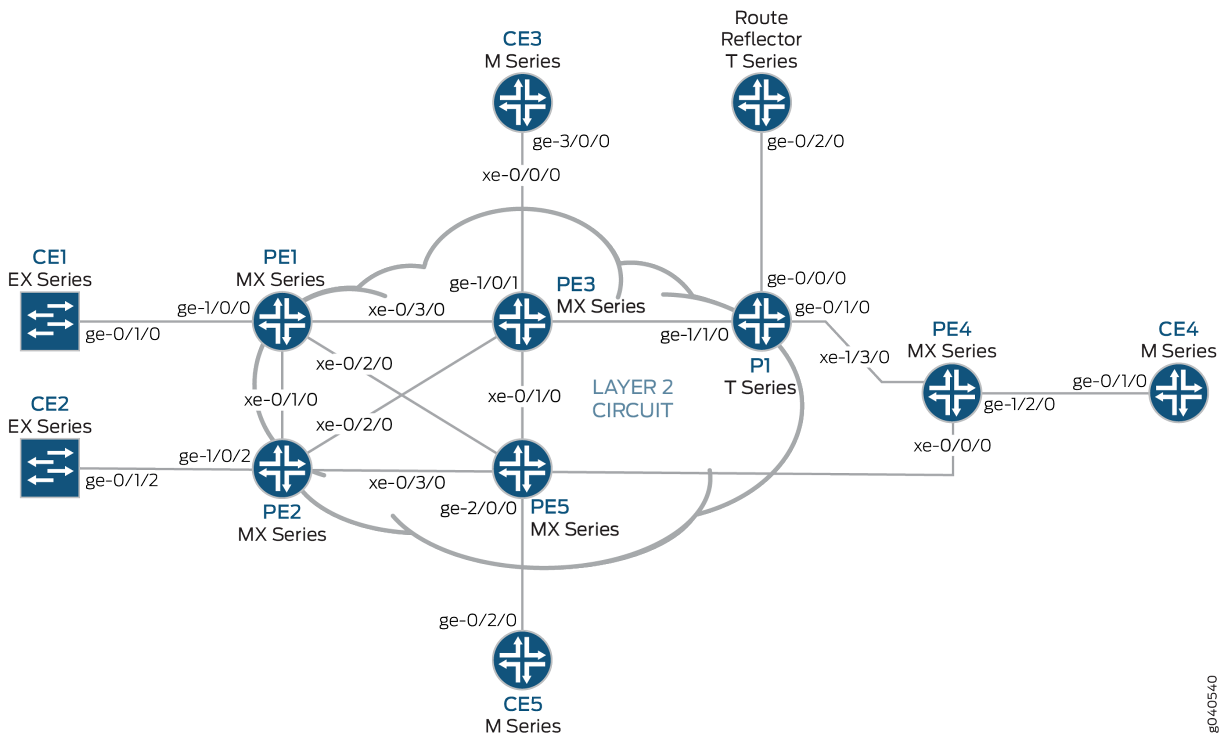
IN THIS SECTION

- Topology | 1271

Topology

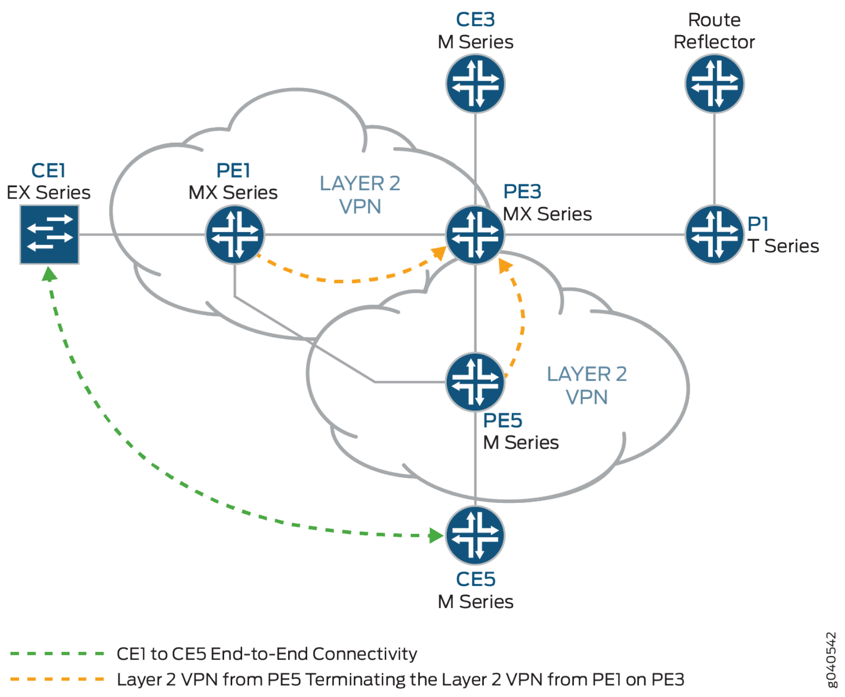
The physical topology of the Layer 2 VPN to Layer 2 VPN connection example is shown in Figure 85 on page 1271.

Figure 85: Physical Topology of a Layer 2 VPN to Layer 2 VPN Connection



The logical topology of a Layer 2 VPN to Layer 2 VPN connection is shown in Figure 86 on page 1272.

Figure 86: Logical Topology of a Layer 2 VPN to Layer 2 VPN Connection



Configuration

IN THIS SECTION

- Configuring Protocols on the PE and P Routers | **1273**
- Verifying the Layer 2 VPN to Layer 2 VPN Connection on Router PE3 | **1279**
- Verifying the Layer 2 VPN to Layer 2 VPN Connection on Router PE3 | **1283**
- Results | **1286**

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the `commit check` command.

In this example, the router being configured is identified using the following command prompts:

- `CE1` identifies the customer edge 1 (CE1) router
- `PE1` identifies the provider edge 1 (PE1) router

- `CE3` identifies the customer edge 3 (CE3) router
- `PE3` identifies the provider edge 3 (PE3) router
- `CE5` identifies the customer edge 5 (CE5) router
- `PE5` identifies the provider edge 5 (PE5) router

This example is organized in the following sections:

Configuring Protocols on the PE and P Routers

Step-by-Step Procedure

All of the PE routers and P routers are configured with OSPF as the IGP protocol. The MPLS, LDP, and BGP protocols are enabled on all of the interfaces except `fxp.0`. Core-facing interfaces are enabled with the MPLS address and inet address.

1. Configure all the PE and P routers with OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except `fxp.0`. The following configuration snippet shows the protocol configuration for Router PE1:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

2. Configure the PE and P routers with OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except `fxp.0`. The following configuration snippet shows the protocol configuration for Router PE3:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.3;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

Step-by-Step Procedure

Configuring the Layer 2 VPN Protocol and Interfaces

1. On Router PE1, configure the `ge-1/0/0` interface encapsulation. To configure the interface encapsulation, include the `encapsulation` statement and specify the `ethernet-ccc` option (vlan-ccc encapsulation is also supported). Configure the `ge-1/0/0.0` logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the `family` statement and specify the `ccc` option. The encapsulation should be configured the same way for all routers in the Layer 2 VPN domain.

```
[edit interfaces]
ge-1/0/0 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.1/24;
        }
    }
}
```

2. On Router PE1, configure the Layer 2 VPN protocols. Configure the remote site ID as 3. Site ID 3 represents Router PE3 (Hub-PE). To configure the Layer 2 VPN protocols, include the `l2vpn` statement

at the `[edit routing-instances *routing-instances-name* protocols]` hierarchy level. Layer 2 VPNs use BGP as the signaling protocol.

```
[edit routing-instances]
L2VPN {
    instance-type l2vpn;
    interface ge-1/0/0.0;
    route-distinguisher 65000:1;
    vrf-target target:65000:2;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site CE1 {
                site-identifier 1;
                interface ge-1/0/0.0 {
                    remote-site-id 3;
                }
            }
        }
    }
}
```

3. On Router PE5, configure the `ge-2/0/0` interface encapsulation by including the `encapsulation` statement and specify the `ethernet-ccc` option. Configure the ge-1/0/0.0 logical interface family for circuit cross-connect functionality by including the `family` statement and specifying the `ccc` option.

```
[edit interfaces]
ge-2/0/0 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.5/24;
        }
    }
}
```

4. On Router PE5, configure the Layer 2 VPN protocols by including the `l2vpn` statement at the `[edit routing-instances` *`routing-instances-name`* `protocols]` hierarchy level. Configure the remote site ID as `3`.

   ```
   [edit routing-instances]
   L2VPN {
       instance-type l2vpn;
       interface ge-2/0/0.0;
       route-distinguisher 65000:5;
       vrf-target target:65000:2;
       protocols {
           l2vpn {
               encapsulation-type ethernet;
               site CE5 {
                   site-identifier 5;
                   interface ge-2/0/0.0 {
                       remote-site-id 3;
                   }
               }
           }
       }
   }
   ```

5. On Router PE3, configure the `iw0` interface with two logical interfaces. To configure the `iw0` interface, include the `interfaces` statement and specify `iw0` as the interface name. For the unit 0 logical interface, include the `peer-unit` statement and specify the logical interface `unit 1` as the peer interface. For the unit 1 logical interface, include the `peer-unit` statement and specify the logical interface `unit 0` as the peer interface.

   ```
   [edit interfaces]
   iw0 {
       unit 0 {
           encapsulation ethernet-ccc;
           peer-unit 1;
       }
       unit 1 {
           encapsulation ethernet-ccc;
           peer-unit 0;
       }
   }
   ```

6. On Router PE3, configure the edge-facing `ge-1/0/1` interface encapsulation by including the `encapsulation` statement and specifying the `ethernet-ccc` option.

```
[edit interfaces]
ge-1/0/1 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
```

7. On Router PE3, configure the logical loopback interface. The loopback interface is used to establish the targeted LDP sessions to Routers PE1 and Router PE5.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
    }
}
```

8. On Router PE3, enable the Layer 2 interworking protocol. To enable the Layer 2 interworking protocol, include the `l2iw` statement at the `[edit protocols]` hierarchy level.

```
[edit protocols]
l2iw;
```

9. On Router PE3, configure two Layer 2 VPN routing instances to terminate the Layer 2 VPN virtual circuits from Router PE1 and Router PE5, as shown.

```
[edit routing-instances]
L2VPN-PE1 {
    instance-type l2vpn;
    interface iw0.0;
    route-distinguisher 65000:3;
    vrf-target target:65000:2;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site CE3 {
                site-identifier 3;
                interface iw0.0 {
                    remote-site-id 1;
                }
            }
        }
    }
}
L2VPN-PE5 {
    instance-type l2vpn;
    interface iw0.1;
    route-distinguisher 65000:33;
    vrf-target target:65000:2;
    protocols {
        l2vpn {
            encapsulation-type ethernet;
            site CE3 {
                site-identifier 3;
                interface iw0.1 {
                    remote-site-id 5;
                }
            }
        }
    }
}
```

Verifying the Layer 2 VPN to Layer 2 VPN Connection on Router PE3

Step-by-Step Procedure

1. BGP is used for control plane signaling in a Layer 2 VPN. On Router PE1, use the `show bgp` command to verify that the BGP control plane for the Layer 2 VPN, has established a neighbor relationship with the route reflector that has IP address `192.0.2.7`.

   Three Layer 2 VPN routes are received from the route reflector for each PE router in the topology.

```
user@PE1> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
```

```
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0            3          3          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.0.2.7               65000        190        192       0       0     1:24:40 Establ
  bgp.l2vpn.0: 3/3/3/0
  L2VPN.l2vpn.0: 3/3/3/0
```

2. On Router PE1, use the `show route` command to verify that the BGP Layer 2 VPN routes are stored in the `L2VPN.l2vpn.0` routing table for each PE router.

```
user@PE1> show route table L2VPN.l2vpn.0
L2VPN.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
65000:1:1:3/96
                   *[L2VPN/170/-101] 01:31:53, metric2 1
                      Indirect
65000:3:3:1/96
                   *[BGP/170] 01:24:58, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.1.2 via xe-0/3/0.0
65000:5:5:3/96
                   *[BGP/170] 01:24:58, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.3.2 via xe-0/2/0.0
65000:33:3:5/96
                   *[BGP/170] 01:24:58, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.1.2 via xe-0/3/0.0
```

3. On Router PE1, use the `show ldp session` command to verify that targeted LDP sessions are established to the PE routers in the network and that the state is `Operational`.

```
user@PE1> show ldp session
  Address           State        Connection     Hold time
192.0.2.2             Operational  Open             24
192.0.2.3             Operational  Open             22
192.0.2.5             Operational  Open             28
```

4. On Router PE1, use the `show l2vpn connections` command to verify that the Layer 2 VPN to site 3 on Router PE3 (Hub-PE) is `Up`.

```
user@PE1> show l2vpn connections

Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy

Legend for interface status
Up -- operational
Dn -- down

Instance: L2VPN
  Local site: CE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    3                         rmt   Up     Jan  5 18:08:25 2010           1
      Remote PE: 192.0.2.3, Negotiated control-word: Yes (Null)
      Incoming label: 800000, Outgoing label: 800000
      Local interface: ge-1/0/0.0, Status: Up, Encapsulation: ETHERNET
    5                         rmt   OR
```

5. On Router PE1, use the `show route` command to verify that the `mpls.0` routing table is populated with the Layer 2 VPN routes used to forward the traffic using an LDP label. Notice that in this example, the router is pushing label `8000000`.

```
user@PE1> show route table mpls.0
[edit]
mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w1d 11:36:44, metric 1
                      Receive
1                  *[MPLS/0] 1w1d 11:36:44, metric 1
                      Receive
2                  *[MPLS/0] 1w1d 11:36:44, metric 1
                      Receive
300432             *[LDP/9] 3d 04:25:02, metric 1
                    > to 10.10.2.2 via xe-0/1/0.0, Pop
300432(S=0)        *[LDP/9] 3d 04:25:02, metric 1
                    > to 10.10.2.2 via xe-0/1/0.0, Pop
300768             *[LDP/9] 3d 04:25:02, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Pop
300768(S=0)        *[LDP/9] 3d 04:25:02, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Pop
300912             *[LDP/9] 3d 04:25:02, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Swap 299856
301264             *[LDP/9] 3d 04:24:58, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Swap 308224
301312             *[LDP/9] 3d 04:25:01, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Pop
301312(S=0)        *[LDP/9] 3d 04:25:01, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Pop
800000             *[L2VPN/7] 01:25:28
                    > via ge-1/0/0.0, Pop       Offset: 4
ge-1/0/0.0         *[L2VPN/7] 01:25:28, metric2 1
                    > to 10.10.1.2 via xe-0/3/0.0, Push 800000 Offset: -4
```

Verifying the Layer 2 VPN to Layer 2 VPN Connection on Router PE3

Step-by-Step Procedure

1. On Router PE3, use the `show l2vpn connections` command to verify that the Layer 2 VPN connections from Router PE1 and Router PE5 are `Up` and are using the `iw0` interface.

```
user@PE3> show l2vpn connections
Instance: L2VPN-PE1
  Local site: CE3 (3)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Jan  5 18:08:22 2010           1
      Remote PE: 192.0.2.1, Negotiated control-word: Yes (Null)
      Incoming label: 800000, Outgoing label: 800000
      Local interface: iw0.0, Status: Up, Encapsulation: ETHERNET
    5                         rmt   OR

Instance: L2VPN-PE5
  Local site: CE3 (3)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   CN
    5                         rmt   Up     Jan  5 18:08:22 2010           1
      Remote PE: 192.0.2.5, Negotiated control-word: Yes (Null)
      Incoming label: 800002, Outgoing label: 800000
      Local interface: iw0.1, Status: Up, Encapsulation: ETHERNET
```

2. On Router PE3, use the `show ldp neighbor` command to verify that the targeted LDP session neighbor IP addresses are shown.

```
user@PE3> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.0.2.1            lo0.0              192.0.2.1:0                44
192.0.2.2            lo0.0              192.0.2.2:0                42
192.0.2.4            lo0.0              192.0.2.4:0                31
192.0.2.5            lo0.0              192.0.2.5:0                44
```

3. On Router PE3, use the `show bgp summary` command to verify that the BGP control plane for the Layer 2 VPN, has established a neighbor relationship with the route reflector that has IP address `192.0.2.7`.

```
user@PE3> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0            2          2          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.0.2.7              65000      10092      10195       0       0  3d 4:23:27 Establ
  bgp.l2vpn.0: 2/2/2/0
  L2VPN-PE1.l2vpn.0: 2/2/2/0
  L2VPN-PE5.l2vpn.0: 2/2/2/0
```

4. On Router PE3, use the `show ldp session` command to verify that targeted LDP sessions are established to all of the PE routers in the network and that the state is `Operational`.

```
user@PE3> show ldp session
Address           State        Connection     Hold time
192.0.2.1             Operational  Open            24
192.0.2.2             Operational  Open            22
192.0.2.4             Operational  Open            20
192.0.2.5             Operational  Open            24
```

5. On Router PE3, use the `show route` command to verify that the `mpls.0` routing table is populated with the Layer 2 VPN routes used to forward the traffic using an LDP label. Notice that in this example, the router is swapping label `800000`. Also notice the two `iw0` interfaces that are used for the Layer 2 interworking routes.

```
user@PE3>show route table mpls.0
mpls.0: 16 destinations, 18 routes (16 active, 2 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w1d 11:50:14, metric 1
                      Receive
1                  *[MPLS/0] 1w1d 11:50:14, metric 1
                      Receive
2                  *[MPLS/0] 1w1d 11:50:14, metric 1
                      Receive
308160             *[LDP/9] 3d 04:38:45, metric 1
```

```
                      > to 10.10.1.1 via xe-0/3/0.0, Pop
308160(S=0)          *[LDP/9] 3d 04:38:45, metric 1
                      > to 10.10.1.1 via xe-0/3/0.0, Pop
308176               *[LDP/9] 3d 04:38:44, metric 1
                      > to 10.10.6.2 via xe-0/1/0.0, Pop
308176(S=0)          *[LDP/9] 3d 04:38:44, metric 1
                      > to 10.10.6.2 via xe-0/1/0.0, Pop
308192               *[LDP/9] 00:07:18, metric 1
                      > to 10.10.20.1 via xe-0/0/0.0, Swap 601649
                        to 10.10.6.2 via xe-0/1/0.0, Swap 299856
308208               *[LDP/9] 3d 04:38:44, metric 1
                      > to 10.10.5.1 via xe-0/2/0.0, Pop
308208(S=0)          *[LDP/9] 3d 04:38:44, metric 1
                      > to 10.10.5.1 via xe-0/2/0.0, Pop
308224               *[LDP/9] 3d 04:38:42, metric 1
                      > to 10.10.20.1 via xe-0/0/0.0, Pop
308224(S=0)          *[LDP/9] 3d 04:38:42, metric 1
                      > to 10.10.20.1 via xe-0/0/0.0, Pop
800000               *[L2IW/6] 01:39:13, metric2 1
                      > to 10.10.6.2 via xe-0/1/0.0, Swap 800000
                      [L2VPN/7] 01:39:13
                      > via iw0.0, Pop       Offset: 4
800002               *[L2IW/6] 01:39:13, metric2 1
                      > to 10.10.1.1 via xe-0/3/0.0, Swap 800000
                      [L2VPN/7] 01:39:13
                      > via iw0.1, Pop       Offset: 4
iw0.0                *[L2VPN/7] 01:39:13, metric2 1
                      > to 10.10.1.1 via xe-0/3/0.0, Push 800000 Offset: -4
iw0.1                *[L2VPN/7] 01:39:13, metric2 1
                      > to 10.10.6.2 via xe-0/1/0.0, Push 800000 Offset: -4
```

Step-by-Step Procedure

Testing Layer 2 VPN to Layer 2 VPN Connectivity (CE1 to CE5)

1. On Router CE1, use the `ping` command to test connectivity to Router CE5. Notice that the response time is in milliseconds, confirming that the ping response is returned.

```
user@CE1>ping 198.51.100.11
PING 198.51.100.11 (198.51.100.11): 56 data bytes
64 bytes from 198.51.100.11: icmp_seq=1 ttl=64 time=22.425 ms
64 bytes from 198.51.100.11: icmp_seq=2 ttl=64 time=1.299 ms
```

```
64 bytes from 198.51.100.11: icmp_seq=3 ttl=64 time=1.032 ms
64 bytes from 198.51.100.11: icmp_seq=4 ttl=64 time=1.029 ms
```

2. On Router CE5, use the `ping` command to test connectivity to Router CE1. Notice that the response time is in milliseconds, confirming that the ping response is returned.

```
user@CE5>ping 198.51.100.1
PING 198.51.100.1 (198.51.100.1): 56 data bytes
64 bytes from 198.51.100.1: icmp_seq=0 ttl=64 time=1.077 ms
64 bytes from 198.51.100.1: icmp_seq=1 ttl=64 time=0.957 ms
64 bytes from 198.51.100.1: icmp_seq=2 ttl=64 time=1.057 ms 1.017 ms
```

Results

The configuration and verification of this example have been completed. The following section is for your reference.

The relevant sample configuration for Router PE1 follows.

Router PE1

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ethernet;
}
    interfaces {
        xe-0/1/0 {
            unit 0 {
                family inet {
                    address 10.10.2.1/30;
                }
                family mpls;
            }
        }
        xe-0/2/0 {
            unit 0 {
                family inet {
                    address 10.10.3.1/30;
                }
                family mpls;
            }
        }
        xe-0/3/0 {
            unit 0 {
                family inet {
                    address 10.10.1.1/30;
                }
                family mpls;
            }
        }
        ge-1/0/0 {
            encapsulation ethernet-ccc;
            unit 0 {
                family ccc;
            }
        }
        lo0 {
            unit 0 {
                family inet {
                    address 192.0.2.1/24;
                }
            }
        }
    }
    routing-options {
        static {
            route 172.16.0.0/8 next-hop 172.19.59.1;
        }
        autonomous-system 65000;
    }
    protocols {
        mpls {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
        bgp {
            group RR {
                type internal;
                local-address 192.0.2.1;
                family l2vpn {
                    signaling;
                }
                neighbor 192.0.2.7;
            }
        }
        ospf {
            traffic-engineering;
            area 0.0.0.0 {
                interface all;
                interface fxp0.0 {
                    disable;
                }
            }
        }
        ldp {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    routing-instances {
        L2VPN {
            instance-type l2vpn;
            interface ge-1/0/0.0;
            route-distinguisher 65000:1;
            vrf-target target:65000:2;
            protocols {
                l2vpn {
                    encapsulation-type ethernet;
                    site CE1 {
                        site-identifier 1;
                        interface ge-1/0/0.0 {
                            remote-site-id 3;
                        }
                    }
                }
            }
        }
    }
```

The relevant sample configuration for Router PE3 follows.

Router PE3

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 3 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ethernet;
}
interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                address 10.10.20.2/30;
            }
            family mpls;
        }
    }
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.6.1/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.2/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.2/30;
            }
            family mpls;
        }
    }
    ge-1/0/1 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    iw0 {
        unit 0 {
            encapsulation ethernet-ccc;
            peer-unit 1;
        }
        unit 1 {
            encapsulation ethernet-ccc;
            peer-unit 0;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.3/24;
            }
        }
    }
}
routing-options {
    static {
        route 172.16.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    l2iw;
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.3;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
routing-instances {
    L2VPN-PE1 {
        instance-type l2vpn;
        interface iw0.0;
        route-distinguisher 65000:3;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                encapsulation-type ethernet;
                site CE3 {
                    site-identifier 3;
                    interface iw0.0 {
                        remote-site-id 1;
                    }
                }
            }
        }
    }
    L2VPN-PE5 {
        instance-type l2vpn;
        interface iw0.1;
        route-distinguisher 65000:33;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                encapsulation-type ethernet;
                site CE3 {
                    site-identifier 3;
                    interface iw0.1 {
                        remote-site-id 5;
                    }
                }
            }
        }
    }
}
```

RELATED DOCUMENTATION

Understanding Layer 2 VPNs | **144**

Layer 2 VPN Applications | **145**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 VPN to a Layer 2 VPN | **1268**

Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview

IN THIS SECTION

- Interconnecting Layer 2 VPNs with Layer 3 VPNs Applications | **1293**

As MPLS-based Layer 2 services grow in demand, new challenges arise for service providers to be able to interoperate with Layer 2 and Layer 3 services and give their customers value-added services. Junos OS has various features to address the needs of service providers. One of these features is the use of a logical tunnel interface. This Junos OS functionality makes use of a tunnel PIC to loop packets out and back from the Packet Forwarding Engine to link the Layer 2 network with the Layer 3 network. The solution is limited by the logical tunnel bandwidth constraints imposed by the tunnel PIC.

Interconnecting Layer 2 VPNs with Layer 3 VPNs Applications

Interconnecting a Layer 2 VPN with a Layer 3 VPN provides the following benefits:

- A single access line to provide multiple services—Traditional VPNs over Layer 2 circuits require the provisioning and maintenance of separate networks for IP and for VPN services. In contrast, Layer 2 VPNs enable the sharing of a provider's core network infrastructure between IP and Layer 2 VPN services, thereby reducing the cost of providing those services.

- Flexibility—Many different types of networks can be accommodated by the service provider. If all sites in a VPN are owned by the same enterprise, this is an intranet. If various sites are owned by different enterprises, the VPN is an extranet. A site can be located in more than one VPN.

- Wide range of possible policies—You can give every site in a VPN a different route to every other site, or you can force traffic between certain pairs of sites routed via a third site and so pass certain traffic through a firewall.

- Scalable network—This design enhances the scalability because it eliminates the need for provider edge (PE) routers to maintain all of the service provider's VPN routes. Each PE router maintains a VRF table for each of its directly connected sites. Each customer connection is mapped to a specific VRF table. Thus, it is a port on the PE router and not a site that is associated with a VRF table. Multiple ports on a PE router can be associated with a single VRF table. It is the ability of PE routers to maintain multiple forwarding tables that supports the per-VPN segregation of routing information.

- Use of route reflectors—Provider edge routers can maintain IBGP sessions to route reflectors as an alternative to a full mesh of IBGP sessions. Deploying multiple route reflectors enhances the scalability of the RFC 2547bis model because it eliminates the need for any single network component to maintain all VPN routes.

- Multiple VPNs are kept separate and distinct from each other—The customer edge routers do not peer with each other. Two sites have IP connectivity over the common backbone only, and only if there is a VPN which contains both sites. This feature keeps the VPNs separate and distinct from each other, even if two VPNs have an overlapping address space.

- Simple for customers to use—Customers can obtain IP backbone services from a service provider, and they do not need to maintain their own backbones.

Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN

IN THIS SECTION

- Requirements | 1294
- Overview and Topology | 1294
- Configuration | 1298
- Verification | 1318

This example provides a step-by-step procedure and commands for interconnecting and verifying a Layer 2 VPN with a Layer 3 VPN. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- Five MX Series routers
- Three M Series routers
- Two T Series routers

Overview and Topology

IN THIS SECTION

- Topology | 1296

A Layer 2 VPN is a type of virtual private network (VPN) that uses MPLS labels to transport data. The communication occurs between the provider edge (PE) routers.

Layer 2 VPNs use BGP as the signaling protocol and, consequently, have a simpler design and require less provisioning overhead than traditional VPNs over Layer 2 circuits. BGP signaling also enables autodiscovery of Layer 2 VPN peers. Layer 2 VPNs can have either a full-mesh or a hub-and-spoke topology. The tunneling mechanism in the core network is, typically, MPLS. However, Layer 2 VPNs can also use other tunneling protocols, such as GRE.

Layer 3 VPNs are based on RFC 2547bis, *BGP/MPLS IP VPNs*. RFC 2547bis defines a mechanism by which service providers can use their IP backbones to provide VPN services to their customers. A Layer 3 VPN is a set of sites that share common routing information and whose connectivity is controlled by a collection of policies. The sites that make up a Layer 3 VPN are connected over a provider's existing public Internet backbone. RFC 2547bis VPNs are also known as BGP/MPLS VPNs because BGP is used to distribute VPN routing information across the provider's backbone, and MPLS is used to forward VPN traffic across the backbone to remote VPN sites.

Customer networks, because they are private, can use either public addresses or private addresses, as defined in RFC 1918, *Address Allocation for Private Internets*. When customer networks that use private addresses connect to the public Internet infrastructure, the private addresses might overlap with the same private addresses used by other network users. MPLS/BGP VPNs solve this problem by adding a *route distinguisher*. A route distinguisher is a VPN identifier prefix that is added to each address from a particular VPN site, thereby creating an address that is unique both within the VPN and within the Internet.

In addition, each VPN has its own VPN-specific routing table that contains the routing information for that VPN only. To separate a VPN's routes from routes in the public Internet or those in other VPNs, the PE router creates a separate routing table for each VPN called a VPN routing and forwarding (VRF) table. The PE router creates one VRF table for each VPN that has a connection to a customer edge (CE) router. Any customer or site that belongs to the VPN can access only the routes in the VRF tables for that VPN. Every VRF table has one or more extended community attributes associated with it that identify the route as belonging to a specific collection of routers. One of these, the *route target* attribute, identifies a collection of sites (VRF tables) to which a PE router distributes routes. The PE router uses the route target to constrain the import of remote routes into its VRF tables.

When an ingress PE router receives routes advertised from a directly connected CE router, it checks the received route against the VRF export policy for that VPN.

- If it matches, the route is converted to VPN-IPv4 format—that is, the route distinguisher is added to the route. The PE router then announces the route in VPN-IPv4 format to the remote PE routers. It also attaches a route target to each route learned from the directly connected sites. The route target attached to the route is based on the value of the VRF table's configured export target policy. The routes are then distributed using IBGP sessions, which are configured in the provider's core network.

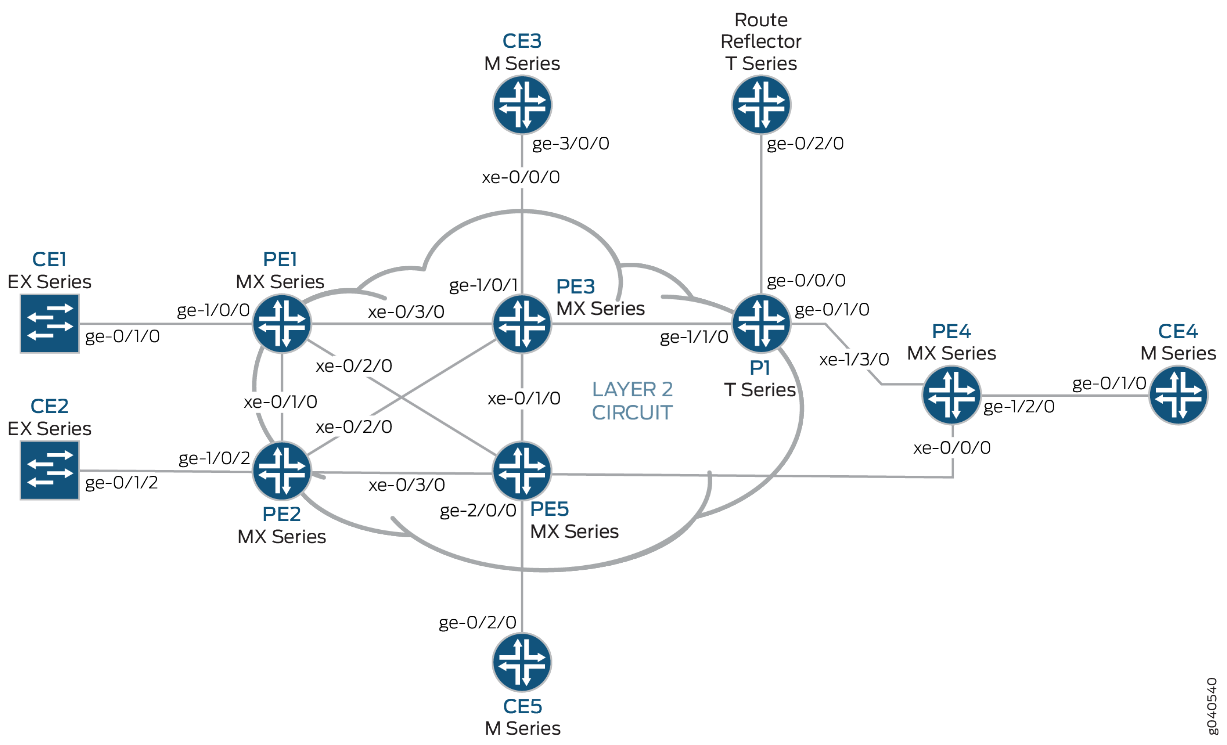
- If the route from the CE router does not match, it is not exported to other PE routers, but it can still be used locally for routing, for example, if two CE routers in the same VPN are directly connected to the same PE router.

When an egress PE router receives a route, it checks it against the import policy on the IBGP session between the PE routers. If it is accepted, the router places the route into its bgp.l3vpn.0 table. At the same time, the router checks the route against the VRF import policy for the VPN. If it matches, the route distinguisher is removed from the route and the route is placed into the VRF table (the *routing-instance-name*.inet.0 table) in IPv4 format.

Topology

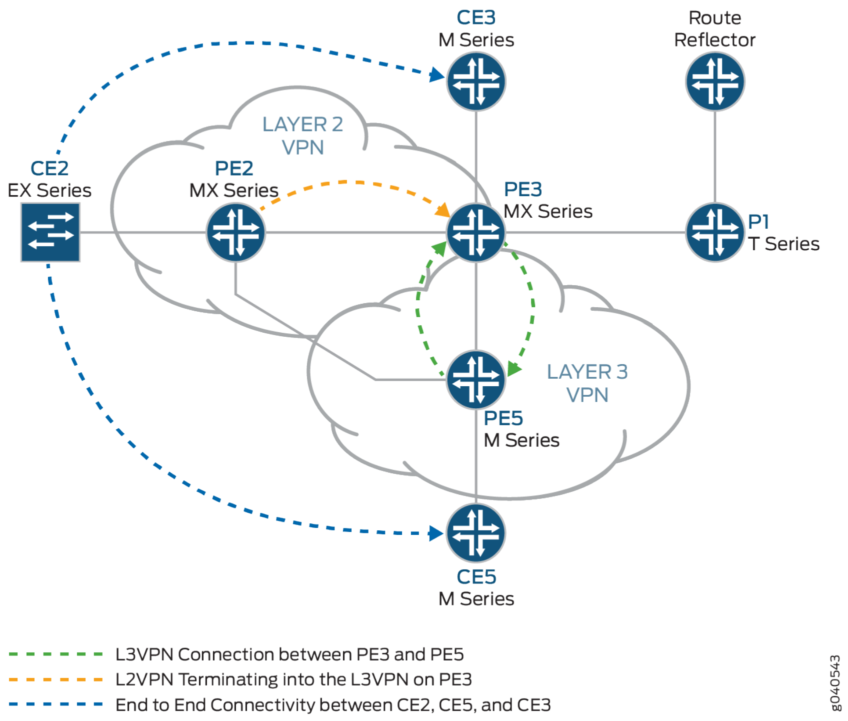
Figure 87 on page 1296 shows the physical topology of a Layer 2 VPN-to-Layer 3 VPN interconnection.

Figure 87: Physical Topology of a Layer 2 VPN Terminating into a Layer 3 VPN



The logical topology of a Layer 2 VPN-to-Layer 3 VPN interconnection is shown in Figure 88 on page 1297.

Figure 88: Logical Topology of a Layer 2 VPN Terminating into a Layer 3 VPN


CE3
M Series
Route
Reflector
LAYER 2
VPN
CE2
EX Series
PE2
MX Series
PE3
MX Series
P1
T Series
LAYER 3
VPN
PE5
M Series
CE5
M Series
L3VPN Connection between PE3 and PE5
L2VPN Terminating into the L3VPN on PE3
End to End Connectivity between CE2, CE5, and CE3
g040543

The following definitions describe the meaning of the device abbreviations used in Figure 87 on page 1296 and Figure 88 on page 1297.

- Customer edge (CE) device—A device at the customer premises that provides access to the service provider’s VPN over a data link to one or more provider edge (PE) routers.

  Typically the CE device is an IP router that establishes an adjacency with its directly connected PE routers. After the adjacency is established, the CE router advertises the site’s local VPN routes to the PE router and learns remote VPN routes from the PE router.

- Provider edge (PE) device—A device, or set of devices, at the edge of the provider network that presents the provider's view of the customer site.

  PE routers exchange routing information with CE routers. PE routers are aware of the VPNs that connect through them, and PE routers maintain VPN state. A PE router is only required to maintain VPN routes for those VPNs to which it is directly attached. After learning local VPN routes from CE routers, a PE router exchanges VPN routing information with other PE routers using IBGP. Finally, when using MPLS to forward VPN data traffic across the provider’s backbone, the ingress PE router functions as the ingress label-switching router (LSR) and the egress PE router functions as the egress LSR.

- Provider (P) device—A device that operates inside the provider's core network and does not directly interface to any CE.

Although the P device is a key part of implementing VPNs for the service provider's customers and may provide routing for many provider-operated tunnels that belong to different VPNs, it is not itself VPN-aware and does not maintain VPN state. Its principal role is allowing the service provider to scale its VPN offerings, for example, by acting as an aggregation point for multiple PE routers.

P routers function as MPLS transit LSRs when forwarding VPN data traffic between PE routers. P routers are required only to maintain routes to the provider's PE routers; they are not required to maintain specific VPN routing information for each customer site.

Configuration

IN THIS SECTION

- Configuring the Base Protocols and Interfaces | **1298**
- Configuring the VPN Interfaces | **1302**

To interconnect a Layer 2 VPN with a Layer 3 VPN, perform these tasks:

Configuring the Base Protocols and Interfaces

Step-by-Step Procedure

1. On each PE and P router, configure OSPF with traffic engineering extensions on all interfaces. Disable OSPF on the fxp0.0 interface.

```
[edit protocols]
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

2. On all the core routers, enable MPLS on all interfaces. Disable MPLS on the fxp0.0 interface.

```
[edit protocols]
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

3. On all the core routers, create an internal BGP peer group and specify the route reflector address (192.0.2.7) as the neighbor. Also enable BGP to carry Layer 2 VPLS network layer reachability information (NLRI) messages for this peer group by including the `signaling` statement at the `[edit protocols bgp group *group-name* family l2vpn]` hierarchy level.

```
[edit protocols]
bgp {
    group RR {
        type internal;
        local-address 192.0.2.2;
        family l2vpn {
            signaling;
        }
        neighbor 192.0.2.7;
    }
}
```

4. On Router PE3, create an internal BGP peer group and specify the route reflector IP address (192.0.2.7) as the neighbor. Enable BGP to carry Layer 2 VPLS NLRI messages for this peer group and enable the processing of VPN-IPv4 addresses by including the `unicast` statement at the `[edit protocols bgp group *group-name* family inet-vpn]` hierarchy level.

```
[edit protocols]
bgp {
    group RR {
        type internal;
        local-address 192.0.2.3;
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        neighbor 192.0.2.7;
    }
}
```

5. For the Layer 3 VPN domain on Router PE3 and Router PE5, enable RSVP on all interfaces. Disable RSVP on the fxp0.0 interface.

```
[edit protocols]
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

6. On Router PE3 and Router PE5, create label-switched paths (LSPs) to the route reflector and the other PE routers. The following example shows the configuration on Router PE5.

```
[edit protocols]
mpls {
    label-switched-path to-RR {
        to 192.0.2.7;
    }
    label-switched-path to-PE2 {
        to 192.0.2.2;
    }
    label-switched-path to-PE3 {
        to 192.0.2.3;
    }
    label-switched-path to-PE4 {
        to 192.0.2.4;
    }
    label-switched-path to-PE1 {
        to 192.0.2.1;
    }
}
```

7. On Routers PE1, PE2, PE3, and PE5, configure the core interfaces with an IPv4 address and enable the MPLS address family. The following example shows the configuration of the xe-0/1/0 interface on Router PE2.

```
[edit]
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.2/30;
            }
            family mpls;
        }
    }
}
```

8. On Router PE2 and Router PE3, configure LDP for the Layer 2 VPN MPLS signaling protocol for all interfaces. Disable LDP on the fxp0.0 interface. (RSVP can also be used.)

```
[edit protocols]
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

9. On the route reflector, create an internal BGP peer group and specify the PE routers IP addresses as the neighbors.

```
[edit]
protocols {
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.7;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            cluster 192.0.2.7;
            neighbor 192.0.2.1;
            neighbor 192.0.2.2;
            neighbor 192.0.2.4;
            neighbor 192.0.2.5;
            neighbor 192.0.2.3;
        }
    }
}
```

10. On the route reflector, configure MPLS LSPs towards Routers PE3 and PE5 to resolve the BGP next hops from inet.3 routing table.

```
[edit]
protocols {
    mpls {
        label-switched-path to-pe3 {
            to 192.0.2.3;
        }

        label-switched-path to-pe5 {
            to 192.0.2.5;
        }
        interface all;
    }
}
```

Configuring the VPN Interfaces

Step-by-Step Procedure

Router PE2 is one end of the Layer 2 VPN. Router PE3 is performing the Layer 2 VPN stitching between the Layer 2 VPN and the Layer 3 VPN. Router PE3 uses the logical tunnel interface (lt interface) configured with different logical interface units applied under two different Layer 2 VPN instances. The packet is looped though the lt interface configured on Router PE3. The configuration of Router PE5 contains the PE-CE interface.

1. On Router PE2, configure the ge-1/0/2 interface encapsulation. Include the encapsulation statement and specify the `ethernet-ccc` option (`vlan-ccc` encapsulation is also supported) at the `[edit interfaces ge-1/0/2]` hierarchy level. The encapsulation should be the same in a whole Layer 2 VPN domain (Routers PE2 and PE3). Also, configure interface lo0.

```
[edit]
interfaces {
    ge-1/0/2 {
        encapsulation ethernet-ccc;
        unit 0;
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.2/24;
            }
        }
    }
}
```

2. On Router PE2, configure the routing instance at the `[edit routing-instances]` hierarchy level. Also, configure the Layer 2 VPN protocol at the `[edit routing-instances` *`routing-instances-name`* `protocols]` hierarchy level. Configure the remote site ID as 3. Site ID 3 represents Router PE3 (Hub-PE). The Layer 2 VPN is using LDP as the signaling protocol. Be aware that in the following example, both the routing instance and the protocol are named `l2vpn`.

```
[edit]
routing-instances {
    l2vpn { # routing instance
        instance-type l2vpn;
        interface ge-1/0/2.0;
        route-distinguisher 65000:2;
        vrf-target target:65000:2;
        protocols {
            l2vpn { # protocol
                encapsulation-type ethernet;
                site CE2 {
                    site-identifier 2;
                    interface ge-1/0/2.0 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

3. On Router PE5, configure the Gigabit Ethernet interface for the PE-CE link `ge-2/0/0` and configure the `lo0` interface.

```
[edit interfaces]
ge-2/0/0 {
    unit 0 {
        family inet {
            address 198.51.100.8/24;
        }
    }
}
lo0 {
    unit 0 {
    }
}
```

4. On Router PE5, configure the Layer 3 VPN routing instance (`L3VPN`) at the `[edit routing-instances]` hierarchy level. Also configure BGP at the `[edit routing-instances L3VPN protocols]` hierarchy level.

```
[edit]
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-2/0/0.0;
        route-distinguisher 65000:5;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                group ce5 {
                    neighbor 198.51.100.2 {
                        peer-as 200;
                    }
                }
            }
        }
    }
}
```

5. In an MX Series router, such as Router PE3, you must create the tunnel services interface to be used for tunnel services. To create the tunnel service interface, include the `bandwidth` statement and specify the amount of bandwidth to reserve for tunnel services in gigabits per second at the `[edit chassis fpc` *`slot-number`* `pic` *`slot-number`* `tunnel-services]` hierarchy level.

```
[edit]
chassis {
    dump-on-panic;
    fpc 1 {
        pic 1 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
}
```

6. On Router PE3, configure the Gigabit Ethernet interface.

   Include the `address` statement at the `[edit interfaces ge-1/0/1.0 family inet]` hierarchy level and specify `198.51.100.9/24` as the IP address.

```
[edit]
interfaces {
    ge-1/0/1 {
        unit 0 {
            family inet {
                address 198.51.100.9/24;
            }
        }
    }
}
```

7. On Router PE3, configure the `lt-1/1/10.0` logical tunnel interface at the `[edit interfaces lt-1/1/10 unit 0]` hierarchy level. Router PE3 is the router that is *stitching* the Layer 2 VPN to the Layer 3 VPN using the logical tunnel interface. The configuration of the peer unit interfaces is what makes the interconnection.

To configure the interface, include the `encapsulation` statement and specify the `ethernet-ccc` option. Include the `peer-unit` statement and specify the logical interface unit `1` as the peer tunnel interface. Include the `family` statement and specify the `ccc` option.

```
[edit]
interfaces {
    lt-1/1/10 {
        unit 0 {
            encapsulation ethernet-ccc;
            peer-unit 1;
            family ccc;
        }
    }
}
```

8. On Router PE3, configure the `lt-1/1/10.1` logical tunnel interface at the `[edit interfaces lt-1/1/10 unit 1]` hierarchy level.

   To configure the interface, include the `encapsulation` statement and specify the `ethernet` option. Include the `peer-unit` statement and specify the logical interface unit `0` as the peer tunnel interface. Include the `family` statement and specify the `inet` option. Include the `address` statement at the `[edit interfaces lt-1/1/10 unit 0]` hierarchy level and specify `198.51.100.7/24` as the IPv4 address.

```
[edit]
interfaces {
    lt-1/1/10 {
        unit 1 {
            encapsulation ethernet;
            peer-unit 0;
            family inet {
                address 198.51.100.7/24;
            }
        }
    }
}
```

9. On Router PE3, add the `lt` interface unit 1 to the routing instance at the `[edit routing-instances L3VPN]` hierarchy level. Configure the instance type as `vrf` with `lt` peer-unit 1 as a PE-CE interface to terminate the Layer 2 VPN on Router PE2 into the Layer 3 VPN on Router PE3.

```
[edit]
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-1/0/1.0;
        interface lt-1/1/10.1;
        route-distinguisher 65000:33;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                export direct;
                group ce3 {
                    neighbor 198.51.100.10 {
                        peer-as 100;
                    }
                }
            }
        }
    }
}
```

10. On Router PE3, add the `lt` interface unit 0 to the routing instance at the `[edit routing-instances protocols l2vpn]` hierarchy level. Also configure the same vrf target for the Layer 2 VPN and Layer 3 VPN routing instances, so that the routes can be leaked between the instances. The example configuration in the previous step shows the vrf target for the `L3VPN` routing instance. The following example shows the vrf target for the `l2vpn` routing instance.

```
[edit]
routing-instances {
    l2vpn {
        instance-type l2vpn;
        interface lt-1/1/10.0;
        route-distinguisher 65000:3;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                encapsulation-type ethernet;
                site CE3 {
                    site-identifier 3;
                    interface lt-1/1/10.0 {
                        remote-site-id 2;
                    }
                }
            }
        }
    }
}
```

11. On Router PE3, configure the `policy-statement` statement to export the routes learned from the directly connected `lt` interface unit 1 to all the CE routers for connectivity, if needed.

```
[edit]
policy-options {
    policy-statement direct {
        term 1 {
            from protocol direct;
            then accept;
        }
    }
}
```

Results

The following output shows the full configuration of Router PE2:

Router PE2

```
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.2/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.4.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/2 {
        encapsulation ethernet-ccc;
        unit 0;
    }
    fxp0 {
        apply-groups [ re0 re1 ];
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.2/24;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.2;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
routing-instances {
    l2vpn {
        instance-type l2vpn;
        interface ge-1/0/2.0;
        route-distinguisher 65000:2;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                encapsulation-type ethernet;
                site CE2 {
                    site-identifier 2;
                    interface ge-1/0/2.0 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

The following output shows the final configuration of Router PE5:

Router PE5

```
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                address 10.10.4.2/30;
            }
            family mpls;
        }
    }
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.6.2/30;
            }
            family mpls;
        }
    }
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.10.9.1/30;
            }
            family mpls;
        }
    }
    xe-1/1/0 {
        unit 0 {
            family inet {
                address 10.10.3.2/30;
            }
            family mpls;
        }
    }
    ge-2/0/0 {
        unit 0 {
            family inet {
                address 198.51.100.8/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.5/24;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all {
            link-protection;
        }
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }

    }
    bgp {
        group to-rr {
            type internal;
            local-address 192.0.2.5;
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-2/0/0.0;
        route-distinguisher 65000:5;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                group ce5 {
                    neighbor 198.51.100.2 {
                        peer-as 200;
                    }
                }
            }
        }
    }
}
```

The following output shows the final configuration of Router PE3:

Router PE3

```
chassis {
    dump-on-panic;
    fpc 1 {
        pic 1 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services ip;
}
interfaces {
    ge-1/0/1 {
        unit 0 {
            family inet {
                address 198.51.100.9/24;
            }
        }
    }
    lt-1/1/10 {
        unit 0 {
            encapsulation ethernet-ccc;
            peer-unit 1;
            family ccc;
        }
        unit 1 {
            encapsulation ethernet;
            peer-unit 0;
            family inet {
                address 198.51.100.7/24;
            }
        }
    }
    xe-2/0/0 {
        unit 0 {
            family inet {
                address 10.10.20.2/30;
            }
            family mpls;
        }
    }
    xe-2/1/0 {
        unit 0 {
            family inet {
                address 10.10.6.1/30;
            }
            family mpls;
        }
    }
    xe-2/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.2/30;
            }
            family mpls;
        }
    }
    xe-2/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.2/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.3/24;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE5 {
            to 192.0.2.5;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.3;
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
policy-options {
    policy-statement direct {
        term 1 {
            from protocol direct;
            then accept;
        }
    }
}
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-1/0/1.0;
        interface lt-1/1/10.1;
        route-distinguisher 65000:33;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                export direct;
                group ce3 {
                    neighbor 198.51.100.10 {
                        peer-as 100;
                    }
                }
            }
        }
    }
    l2vpn {
        instance-type l2vpn;
        interface lt-1/1/10.0;
        route-distinguisher 65000:3;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                encapsulation-type ethernet;
                site CE3 {
                    site-identifier 3;
                    interface lt-1/1/10.0 {
                        remote-site-id 2;
                    }
                }
            }
        }
    }
}
```

Verification

IN THIS SECTION

- Verifying Router PE2 VPN Interface | **1319**
- Verifying Router PE3 VPN Interface | **1321**
- Verifying End-to-End connectivity from Router CE2 to Router CE5 and Router CE3 | **1324**

Verify the Layer 2 VPN-to-Layer 3 VPN interconnection:

Verifying Router PE2 VPN Interface

Purpose

Check that the Layer 2 VPN is up and working at the Router PE2 interface and that all the routes are there.

Action

1. Use the `show l2vpn connections` command to verify that the connection site ID is 3 for Router PE3 and that the status is `Up`.

```
user@PE2> show l2vpn connections
Layer-2 VPN connections:
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby

Legend for interface status
Up -- operational
Dn -- down

Instance: l2vpn
  Local site: CE2 (2)
    connection-site     Type  St     Time last up        # Up trans
    3                   rmt   Up     Jan  7 14:14:37 2010         1
      Remote PE: 192.0.2.3, Negotiated control-word: Yes (Null)
```

```
        Incoming label: 800000, Outgoing label: 800001
        Local interface: ge-1/0/2.0, Status: Up, Encapsulation: ETHERNET
```

2. Use the `show route table` command to verify that the Layer 2 VPN route is present and that there is a next hop of `10.10.5.2` through the `xe-0/2/0.0` interface. The following output verifies that the Layer 2 VPN routes are present in the l2vpn.l2vpn.0 table. Similar output should be displayed for Router PE3.

```
user@PE2> show route table l2vpn.l2vpn.0
l2vpn.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

65000:2:2:3/96
                   *[L2VPN/170/-101] 02:40:35, metric2 1
                      Indirect
65000:3:3:1/96
                   *[BGP/170] 02:40:35, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.5.2 via xe-0/2/0.0
```

3. Verify that Router PE2 has a Layer 2 VPN MPLS label pointing to the LDP label to Router PE3 in both directions (PUSH and POP).

```
user@PE2> show route table mpls.0
 mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
300560             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
300560(S=0)        *[LDP/9] 19:45:53, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
301008             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Swap 299856
301536             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301536(S=0)        *[LDP/9] 19:45:53, metric 1
```

```
                     > to 10.10.4.2 via xe-0/3/0.0, Pop
301712              *[LDP/9] 16:14:52, metric 1
                     > to 10.10.5.2 via xe-0/2/0.0, Swap 315184
301728              *[LDP/9] 16:14:52, metric 1
                     > to 10.10.5.2 via xe-0/2/0.0, Pop
301728(S=0)         *[LDP/9] 16:14:52, metric 1
                     > to 10.10.5.2 via xe-0/2/0.0, Pop
800000              *[L2VPN/7] 02:40:35
                     > via ge-1/0/2.0, Pop       Offset: 4
ge-1/0/2.0          *[L2VPN/7] 02:40:35, metric2 1
                     > to 10.10.5.2 via xe-0/2/0.0, Push 800001 Offset: -4
```

Meaning

The `l2vpn` routing instance is up at interface `ge-1/0/2` and the Layer 2 VPN route is shown in table l2vpn.l2vpn.0. Table `mpls.0` shows the Layer 2 VPN routes used to forward the traffic using an LDP label.

Verifying Router PE3 VPN Interface

Purpose

Check that the Layer 2 VPN connection from Router PE2 and Router PE3 is `Up` and working.

Action

1. Verify that the BGP session with the route reflector for the family `l2vpn-signaling` and the family `inet-vpn` is established.

```
user@PE3> show bgp summary
Groups: 2 Peers: 2 Down peers: 0
Table        Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0          1          1          0          0          0          0
bgp.L3VPN.0          1          1          0          0          0          0
Peer         AS    InPkt     OutPkt    OutQ   Flaps Last Up/Dwn   State|#Active /Received/
Accepted/Damped...
192.0.2.7  65000    2063      2084      0       1    15:35:16   Establ
  bgp.l2vpn.0: 1/1/1/0
  bgp.L3VPN.0: 1/1/1/0
  L3VPN.inet.0: 1/1/1/0
  l2vpn.l2vpn.0: 1/1/1/0
```

2. The following output verifies the Layer 2 VPN route and the label associated with it.

```
user@PE3> show route table l2vpn.l2vpn.0 detail
l2vpn.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
65000:2:2:3/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65000:2
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 192.0.2.7
                Protocol next hop: 192.0.2.2
                Indirect next hop: 2 no-forward
                State: <Secondary Active Int Ext>
                Local AS: 65000 Peer AS: 65000
                Age: 2:45:52    Metric2: 1
                Task: BGP_65000.192.0.2.7+60585
                Announcement bits (1): 0-l2vpn-l2vpn
                AS path: I (Originator) Cluster list:  192.0.2.7
                AS path:  Originator ID: 192.0.2.2
                Communities: target:65000:2 Layer2-info: encaps:ETHERNET, control
flags:Control-Word, mtu: 0, site preference: 100  Accepted
                Label-base: 800000, range: 2, status-vector: 0x0
                Localpref: 100
                Router ID: 192.0.2.7
                Primary Routing Table bgp.l2vpn.0
```

3. The following output show the L2VPN MPLS.0 route in the mpls.0 route table.

```
user@PE3> show route table mpls.0
mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
16                 *[VPN/0] 15:59:24
                      to table L3VPN.inet.0, Pop
315184             *[LDP/9] 16:21:53, metric 1
```

```
                      > to 10.10.20.1 via xe-2/0/0.0, Pop
315184(S=0)          *[LDP/9] 16:21:53, metric 1
                      > to 10.10.20.1 via xe-2/0/0.0, Pop
315200               *[LDP/9] 01:13:44, metric 1
                        to 10.10.20.1 via xe-2/0/0.0, Swap 625297
                      > to 10.10.6.2 via xe-2/1/0.0, Swap 299856
315216               *[LDP/9] 16:21:53, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, Pop
315216(S=0)          *[LDP/9] 16:21:53, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, Pop
315232               *[LDP/9] 16:21:45, metric 1
                      > to 10.10.1.1 via xe-2/3/0.0, Pop
315232(S=0)          *[LDP/9] 16:21:45, metric 1
                      > to 10.10.1.1 via xe-2/3/0.0, Pop
315248               *[LDP/9] 16:21:53, metric 1
                      > to 10.10.5.1 via xe-2/2/0.0, Pop
315248(S=0)          *[LDP/9] 16:21:53, metric 1
                      > to 10.10.5.1 via xe-2/2/0.0, Pop
315312               *[RSVP/7] 15:02:40, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
315312(S=0)          *[RSVP/7] 15:02:40, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
315328               *[RSVP/7] 15:02:40, metric 1
                      > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
315360               *[RSVP/7] 15:02:40, metric 1
                      > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
316272               *[RSVP/7] 01:13:27, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, label-switched-path Bypass->10.10.9.1
316272(S=0)          *[RSVP/7] 01:13:27, metric 1
                      > to 10.10.6.2 via xe-2/1/0.0, label-switched-path Bypass->10.10.9.1
800001               *[L2VPN/7] 02:47:33
                      > via lt-1/1/10.0, Pop       Offset: 4
lt-1/1/10.0          *[L2VPN/7] 02:47:33, metric2 1
                      > to 10.10.5.1 via xe-2/2/0.0, Push 800000 Offset: -4
```

4. Use the `show route table mpls.0` command with the `detail` option to see the BGP attributes of the route such as next-hop type and label operations.

```
user@PE5> show route table mpls.0 detail
lt-1/1/10.0 (1 entry, 1 announced)
        *L2VPN  Preference: 7
                Next hop type: Indirect
                Next-hop reference count: 2
                Next hop type: Router, Next hop index: 607
                Next hop: 10.10.5.1 via xe-2/2/0.0, selected
                Label operation: Push 800000 Offset: -4
                Protocol next hop: 192.0.2.2
                Push 800000 Offset: -4
                Indirect next hop: 8cae0a0 1048574
                State:  <Active Int>
                Age: 2:46:34    Metric2: 1
                Task: Common L2 VC
                Announcement bits (2): 0-KRT 2-Common L2 VC
                AS path: I
                Communities: target:65000:2 Layer2-info: encaps:ETHERNET, control
flags:Control-Word, mtu: 0, site preference: 100
```

Verifying End-to-End connectivity from Router CE2 to Router CE5 and Router CE3

Purpose

Check the connectivity between Routers CE2, CE3, and CE5.

Action

1. Ping the Router CE3 IP address from Router CE2.

```
user@CE2> ping 198.51.100.10 # CE3 IP address
PING 198.51.100.10 (198.51.100.10): 56 data bytes
64 bytes from 198.51.100.10: icmp_seq=0 ttl=63 time=0.708 ms
64 bytes from 198.51.100.10: icmp_seq=1 ttl=63 time=0.610 ms
```

2. Ping the Router CE5 IP address from Router CE2.

```
user@CE2> ping 198.51.100.2 # CE5 IP address
PING 198.51.100.2 (198.51.100.2): 56 data bytes
64 bytes from 198.51.100.2: icmp_seq=0 ttl=62 time=0.995 ms
64 bytes from 198.51.100.2: icmp_seq=1 ttl=62 time=1.005 ms
```

Configure an Anycast Gateway in an L2VPN

SUMMARY

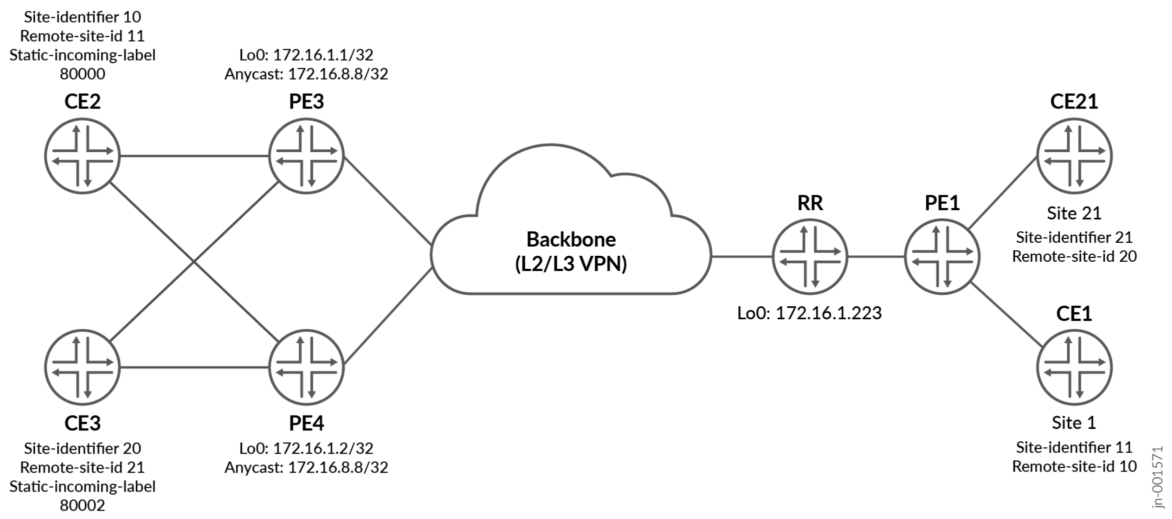
Configure a single IP address to function as an anycast gateway across multiple devices within a L2VPN.

Anycast is a networking feature where you can assign the same IP address to multiple devices in a network. Anycast enables the devices to use the same IP address as a default gateway for incoming traffic. The benefits of anycast include redundant connectivity and improved resiliency in the network.

In an MPLS network, PE devices typically assign unique inner VPN labels dynamically to identify the different VPN. The outer label is used to forward the packet across the service provider's core network, while the inner VPN label is used to identify a specific customer VPN. When you have multiple devices sharing an IP address for the gateway, dynamic label allocation can lead to inconsistencies in the VPN label. This causes duplicate entries in the routing table and inefficiencies in the network. Starting in Junos OS Release 25.4R1, we support static labels for each VPN site.

Figure 89 on page 1326 illustrates a topology of a L2VPN with an anycast IP address. The PE devices (PE3 and PE4) have different loopback addresses for the BGP protocol, but share an IP address for the anycast gateway. The sample configuration for configuring anycast address on PE3 is described below. Use a similar configuration for PE4.

Figure 89: Anycast Gateway in an L2VPN



1. Configure the interfaces. Assign one loopback address that can be used by as an identifier by BGP and another loopback address that can be used as an anycast IP address.

```
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces lo0.0 family inet address 172.16.8.8/32
```

2. Configure the policy statement to use the anycast IP address as the source for VPN traffic.

```
set policy-options policy-statement nhs-to-core-vpn term 1 then next-hop 172.16.8.8/32
set policy-options policy-statement nhs-to-core-vpn term 1 then accept
```

3. Configure the BGP protocol to support L2VPN routing. Configure the route reflector as a neighbor and apply the export policy

```
set protocols bgp group iBGP-VPN-RR description "to VPN Route Reflector"
set protocols bgp group iBGP-VPN-RR local-address 172.16.1.1
set protocols bgp group iBGP-VPN-RR family inet-vpn unicast
set protocols bgp group iBGP-VPN-RR family l2vpn signaling
set protocols bgp group iBGP-VPN-RR export nhs-to-core-vpn
set protocols bgp group iBGP-VPN-RR neighbor 172.16.1.223 description VPN-RR
set protocols bgp group iBGP-VPN-RR vpn-apply-export
```

4. Configure an L2VPN routing instance with a unique static label for each local VPN site (CE2 and CE3).

NOTE: The static label can be in the range of 16 through 983,615 in increments of 2. In other words, you cannot assign consecutive numbers for the static label.

```
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE2 site-identifier 10
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE2 interface
et-0/0/0.1 remote-site-id 11
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE2 static incoming-
label 80000
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE3 site-identifier 20
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE3 interface
et-0/0/0.2 remote-site-id 21
set routing-instances L2VPN1 instance-type l2vpn protocols l2vpn site CE3 static incoming-
label 80002
```

Use the **show l2vpn connections** command to verify that the labels are being applied.

```
regress@PE1_re> show l2vpn connections
Instance: L2VPN1
Edge protection: Not-Primary
  Local site: CE2 (10)
connection-site           Type  St     Time last up          # Up trans
11                      rmt   Up     Apr  9 13:51:11 2025           1
      Remote PE: 172.16.7.7, Negotiated control-word: Yes (Null)
      Incoming label: 800000, Outgoing label: 800009
      Local interface: et-0/0/0.1, Status: Up, Encapsulation: VLAN
      Flow Label Transmit: No, Flow Label Receive: No
 regress@PE1_re> show route table mpls.0
mpls.0: 13 destinations, 15 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
    800000      [L2VPN/7] 23:46:00
                  >  via et-0/0/0.1, Pop     Offset: 4.
et-0/0/0.1   *[L2VPN/7] 23:46:00, metric2 1
                  >  to 2.0.0.1 via et-0/0/0.0, Push 800009, Push 299792(top) Offset: 252
```

Stitching Layer 2 VPNs and VPWS

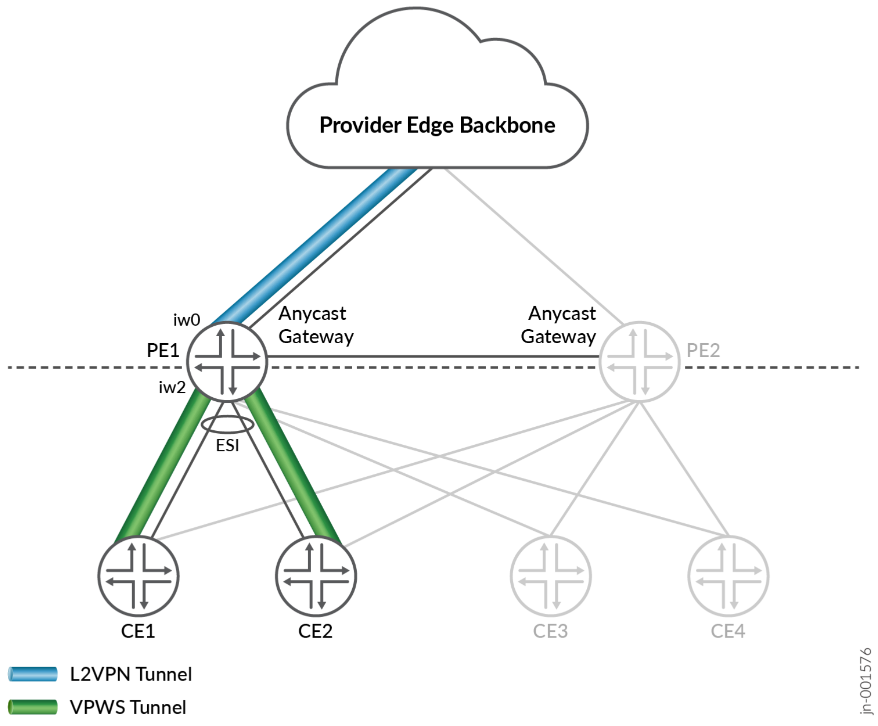
SUMMARY

Stitching an Layer 2 VPN (L2VPN) and a Virtual Private Wire Service (VPWS) tunnel enables seamless forwarding of traffic in the network. Configure interwork interfaces to stitch the L2VPN and VPWS tunnel without a physical interface.

When two Layer 2 point-to-point services converge on a device, you can stitch the two services together to move traffic across seamlessly. In a spine-leaf architecture, the spine acts as an aggregation point where southbound traffic from the backbone uses an L2VPN and northbound traffic from the access layer uses VPWS. Stitching allows you to stitch an L2VPN tunnel to a VPWS tunnel seamlessly without dedicating a physical hardware interface to loop packets from the L2VPN to VPWS.

Figure 90 on page 1329 shows a spine-leaf topology where the spines connect to the backbone of a provider's network and the leaf devices connect to the customer's access network. The spine is the convergence point for the L2VPN and VPWS.

Figure 90: Spine-Leaf Topology with L2VPN and VPWS Tunnels



The following sample configuration snippet is for PE1. You must use a similar configuration for PE2.

1. Set up the interwork interfaces for the two Layer 2 services that will be peered.

```
set interfaces iw0 unit 0
set interfaces iw2 unit 0
```

2. Configure the ESI on the logical interface facing the access layer. This allows leaf devices to send packets to either spine.

```
set interfaces iw2 esi 00:01:02:03:04:05:06:07:08:09
set interfaces iw2 esi all-active
```

3. Configure the L2VPN routing instance. Identify the associate peering instance that will be stitched. For this example, VPWS1 is the peer instance that will be stitched to L2VPN1. We specify CE1 as our local VPN site and configure a static incoming label of 200 to utilize an anycast gateway. For

more information about anycast gateways in an L2VPN, see "Configure an Anycast Gateway in an L2VPN" on page 1325.

```
set routing-instances L2VPN1 protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN1 protocols l2vpn site CE1 site-identifier 1
set routing-instances L2VPN1 protocols l2vpn site CE1 interface iw0.0 remote-site-id 2
set routing-instances L2VPN1 protocols l2vpn site CE1 interface iw0.0 incoming-label static 
200
set routing-instances L2VPN1 vpws-peer-instance VPWS1
```

4. Configure the VPWS routing instance. We will use EVPN signaling and configure an EVPN-VPWS instance type. Identify the associate peering instance that will be stitched. L2VPN1 is the peer instance that will be stitched to VPWS1.

```
set routing-instances VPWS1 protocols evpn interface iw2.0 vpws-service-id local 1
set routing-instances VPWS1 protocols evpn interface iw2.0 vpws-service-id remote 2
set routing-instances VPWS1 protocols evpn l2vpn-peer-instance L2VPN1
```

5. Enable `dynamic-peer-discovery` under the Layer 2 Interwork protocol. When this feature is enabled, the device matches the site identifier (`site-identifier`) configured under the Layer 2 VPN routing instance with the VPWS service identifier (`local vpws-service-id`) under the VPWS routing instance and pairs the associated interfaces as peers.

```
set protocols l2iw dynamic-peer-discover
```

6. Enable `control-word` for both the `l2vpn` and `evpn` instances to ensure that the PE device processes data encapsulation consistently.

```
set routing-instances VPWS1 protocols epvn control-word
set routing-instancee L2VPN1 protocols l2vpn control-word
```

7. Enable composite next hop for improved convergence in the L2VPN.

```
set routing-options forwarding-table chained-composite-next-hop ingress l2vpn
```

NOTE: Enable load balancing on the leaf devices to ensure that the traffic will be distributed across the spines.

```
set policy-options policy-statement pplb then load-balance per-packet
set routing-options forwarding-table export pplb
```

Verification

Confirm that the L2VPN and VPWS tunnels are stitched using the following commands:

- show l2vpn connections
- show evpn vpws-instance
- show route table mpls.0

Use the `show l2vpn connections` command to display the label used in the VPN connection.

```
user@pe1>show l2vpn connections
Edge protection: Not-Primary
          Local site: CE1 (10)
             connection-site           Type  St     Time last up          # Up trans
             11                      rmt   Up     May 12 12:11:51 2025           1
             Remote PE: 172.16.7.7, Negotiated control-word: Yes (Null)
           Incoming label: 200, Outgoing label: 100
                 Local interface: iw0.0, Status: Up, Encapsulation: VLAN
                 Flow Label Transmit: No, Flow Label Receive: No
```

Use the `show evpn vpws-instance` command to display label used in the VPWS instance.

```
user@pe1> show evpn vpws-instance
     Instance: VPWS1, Instance type: EVPN VPWS, Encapsulation type: MPLS, Enhanced convergence:
Disabled
     Route Distinguisher: 172.16.1.1:10
     Number of local interfaces: 2 (0 up)
         Interface name  ESI                            Mode          Role       Status
Control-Word    Flow-Label-Tx    Flow-Label-Rx
         iw2.0          00:01:02:03:04:05:06:07:08:09 all-active      Primary    Up
No              No               No
```

```
          Local SID: 10 Advertised Label: 300
          Remote SID: 11
          PE addr         ESI                          Label  End.Dx2 SID     Mode
Role     TS                       Status
            172.16.1.4    (LEAF)     00:00:00:00:00:00:00:00:00:00 400                single-
homed   Primary  2025-05-12 12:06:55.598 Resolved
```

Use the `show route table mpls.0` command to verify that packet labels are being swapped

```
user@pe1>show route table mpls.0
    200             *[L2IW/6] 00:11:33, metric2 1
                          >  to 4.0.0.1 via et-0/0/3.0, Swap 400.
                         [L2VPN/7] 00:11:33
                          >  via iw2.0, Pop       Offset: 4
   300             *[L2IW/6] 00:11:33, metric2 1
                          >  to 7.0.0.2 via et-0/0/0.0, Swap 100, Push 299776(top)
            [EVPN/7] 00:12:24
                     >  via iw0.0, Pop
```

CHAPTER 43

Connecting Layer 2 Circuits to Other VPNs

IN THIS CHAPTER

- Using the Layer 2 Interworking Interface to Interconnect a Layer 2 Circuit to a Layer 2 VPN | **1333**
- Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN | **1335**
- Example: Interconnecting a Layer 2 Circuit with a Layer 2 Circuit | **1346**
- Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN | **1368**
- Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN | **1369**

Using the Layer 2 Interworking Interface to Interconnect a Layer 2 Circuit to a Layer 2 VPN

Instead of using a physical Tunnel PIC for looping the packet received from the Layer 2 circuit, the Layer 2 interworking interface uses Junos OS to stitch together both Layer 2 VPN routes.

To configure the interworking interface, include the **iw0** statement. The **iw0** statement is configured at the **[edit interfaces]** hierarchy level. This specifies the peering between two logical interfaces. This configuration is similar to the configuration for a logical tunnel interface. The logical Interfaces must be associated with the endpoints of a Layer 2 circuit and Layer 2 VPN connections.

```
[edit interfaces]
iw0 {
    unit 0 {
        peer-unit 1;
    }
    unit 1 {
        peer-unit 0;
    }
}
```

Configure the Layer 2 circuit protocol by including the `l2circuit` statement at the `[edit protocols]` hierarchy level and specifying the `neighbor` and `iw0` interface.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.0 {
        interface iw0.0;
    }
}
```

Configure the Layer 2 VPN connection, by including the *`routing-instance-name`* statement at the `[edit routing-instances]` hierarchy level and specifying the `instance-type l2vpn` option.

```
[edit routing-instances]
routing-instance-name {
    instance-type l2vpn;
    interface iw0.1;
    ...
    protocols {
        l2vpn {
            <l2vpn configuration>;
        }
    }
}
```

In addition to the `iw0` interface configuration, Layer 2 interworking `l2iw` protocols must be enabled. Without the `l2iw` configuration, the `l2iw` routes will not be formed, regardless of whether any `iw` interfaces are present. Within the l2iw protocols, only trace options can be configured in the standard fashion. The minimum configuration necessary for the feature to work is shown below:

```
[edit]
protocols {
    l2iw;
}
```

RELATED DOCUMENTATION

Layer 2 Circuit Overview | **351**

Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN | **1335**

Example: Interconnecting a Layer 2 Circuit with a Layer 2 VPN

IN THIS SECTION

- Requirements | 1335
- Overview and Topology | 1335
- Configuration | 1337

This example provides a step-by-step procedure and commands for configuring and verifying a Layer 2 circuit to a Layer 2 VPN. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- 2 MX Series 5G Universal Routing Platforms
- 2 M Series Multiservice Edge Router
- 1 T Series Core Router
- 1 EX Series Ethernet Switch

Overview and Topology

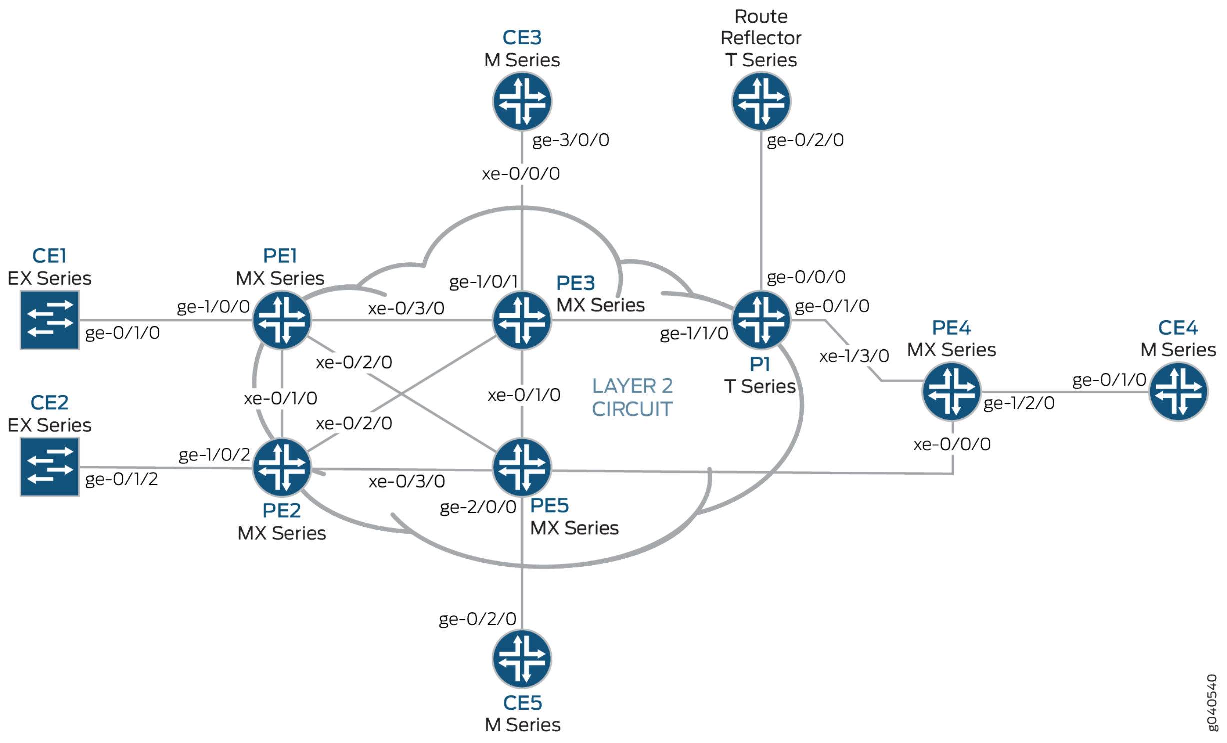
IN THIS SECTION

- Topology | 1335

Topology

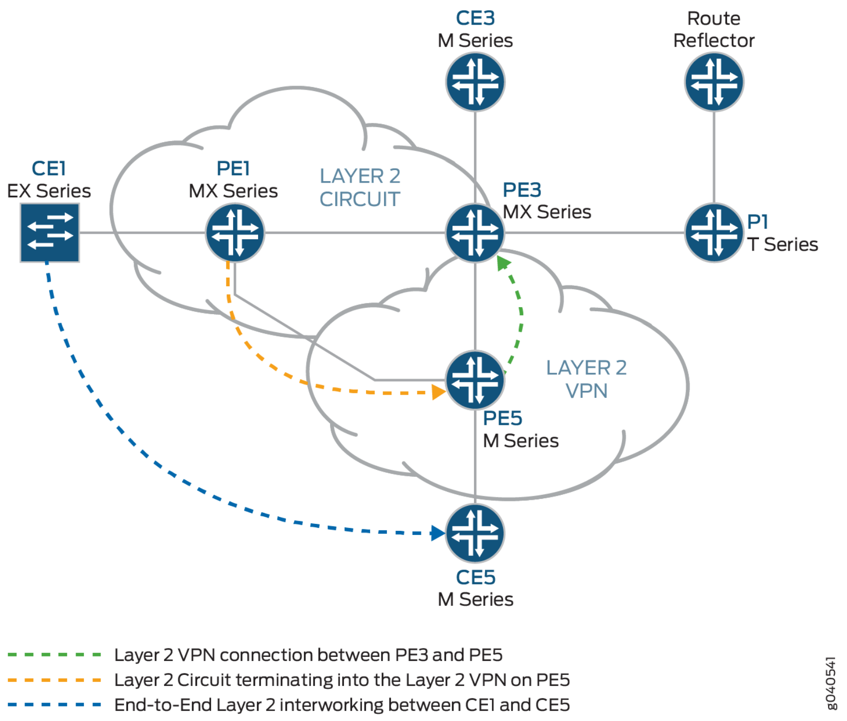
The physical topology of a Layer 2 circuit to a Layer 2 VPN connection is shown in Figure 91 on page 1336.

Figure 91: Physical Topology of a Layer 2 Circuit to a Layer 2 VPN Connection



The logical topology of a Layer 2 circuit to a Layer 2 VPN connection is shown in Figure 92 on page 1337.

Figure 92: Logical Topology of a Layer 2 Circuit to a Layer 2 VPN Connection


CE3
M Series
Route
Reflector
CE1
EX Series
PE1
MX Series
LAYER 2
CIRCUIT
PE3
MX Series
P1
T Series
LAYER 2
VPN
PE5
M Series
CE5
M Series
Layer 2 VPN connection between PE3 and PE5
Layer 2 Circuit terminating into the Layer 2 VPN on PE5
End-to-End Layer 2 interworking between CE1 and CE5
g040541

Configuration

IN THIS SECTION

- Configuring Protocols on the PE and P Routers | **1338**
- Verification | **1344**

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the `commit check` command.

In this example, the router being configured is identified using the following command prompts:

- `CE1` identifies the customer edge 1 (CE1) router
- `PE1` identifies the provider edge 1 (PE1) router
- `CE3` identifies the customer edge 3 (CE3) router

- `PE3` identifies the provider edge 3 (PE3) router
- `CE5` identifies the customer edge 5 (CE5) router
- `PE5` identifies the provider edge 5 (PE5) router

This example is organized in the following sections:

Configuring Protocols on the PE and P Routers

Step-by-Step Procedure

In this example, all of the PE routers and P routers are configured with OSPF as the IGP protocol. The MPLS, LDP, and BGP protocols are enabled on all of the interfaces except **fxp.0**. Core-facing interfaces are enabled with the MPLS address and inet address.

1. Configure all the PE and P routers with OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except `fxp.0`. LDP is used as the signaling protocol on Router PE1 for the Layer 2 circuit. The following configuration snippet shows the protocol configuration for Router PE1:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.1;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

2. Configure the PE and P routers with OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except `fxp.0`. BGP is used as the signaling protocol on Router PE3 for the Layer 2 VPN. The following configuration snippet shows the protocol configuration for Router PE3:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.3;
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

Step-by-Step Procedure

Configuring Interfaces

1. On Router PE1, configure the `ge-1/0/0` interface encapsulation. To configure the interface encapsulation, include the `encapsulation` statement and specify the `ethernet-ccc` option (vlan-ccc encapsulation is also supported). Configure the `ge-1/0/0.0` logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the `family` statement and specify the `ccc` option. The encapsulation should be configured the same way for all routers in the Layer 2 circuit domain.

   ```
   [edit interfaces]
   ge-1/0/0 {
       encapsulation ethernet-ccc;
       unit 0 {
           family ccc;
       }
   }
   lo0 {
       unit 0 {
           family inet {
               address 192.0.2.1/24;
           }
       }
   }
   ```

2. Router PE5 is the router that is *stitching* the Layer 2 circuit to the Layer 2 VPN using the interworking interface. The configuration of the peer unit interfaces is what makes the interconnection.

   On Router PE5, configure the `iw0` interface with two logical interfaces. To configure the `iw0` interface, include the `interfaces` statement and specify `iw0` as the interface name. For the unit 0 logical interface, include the `peer-unit` statement and specify the logical interface `unit 1` as the peer interface. For the unit 1 logical interface, include the `peer-unit` statement and specify the logical interface `unit 0` as the peer interface.

```
[edit interfaces]
iw0 {
    unit 0 {
        encapsulation ethernet-ccc;
        peer-unit 1;
    }
    unit 1 {
        encapsulation ethernet-ccc;
        peer-unit 0;
    }
}
```

3. On Router PE5, configure the logical loopback interface. The loopback interface is used to establish the targeted LDP sessions to Routers PE1 and PE5.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.5/24;
        }
    }
}
```

Step-by-Step Procedure

Configuring the Layer 2 circuit protocol

1. On Router PE1, configure the IP address of the remote PE router with the `neighbor` statement. The loopback address and router ID of the PE neighbor is commonly the neighbor’s IP address. To allow a Layer 2 circuit to be established even though the maximum transmission unit (MTU) configured on the PE router does not match the MTU configured on the remote PE router, include the `ignore-mtu-mismatch` statement.

```
[edit]
protocols {
    l2circuit {
        neighbor 192.0.2.5 {
            interface ge-1/0/0.0 {
                virtual-circuit-id 100;
                no-control-word;
                ignore-mtu-mismatch;
            }
        }
    }
}
```

2. On Router PE5, configure the IP address of the remote PE router. To configure the IP address of the remote PE router, include the `neighbor` statement and specify the IP address of the loopback interface on Router PE1. Configure the virtual circuit ID to be the same as the virtual circuit ID on the neighbor router. To allow a Layer 2 circuit to be established even though the MTU configured on the local PE router does not match the MTU configured on the remote PE router, include the `ignore-mtu-mismatch` statement. Also disable the use of the control word for demultiplexing by including the `no-control-word` statement.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.1 {
        interface iw0.0 {
            virtual-circuit-id 100;
            no-control-word;
            ignore-mtu-mismatch;
        }
    }
}
```

3. On Router PE5, configure the Layer 2 VPN protocols by including the `l2vpn` statement at the `[edit routing-instances` *`routing-instances-name`* `protocols]` hierarchy level. To configure the `iw0` interface, include the `interfaces` statement and specify `iw0` as the interface name. The `iw0` interface is configured under the Layer 2 VPN protocols to receive the looped packet from the `iw0.1` logical interface. The `l2vpn` protocol is configured on Router PE5 with site CE5, which is configured in the BGP L2VPN routing instance. Router CE1 has communication to Router CE5, through the Layer 2 interworking configuration on Router PE5.

```
[edit]
routing-instances {
    L2VPN {
        instance-type l2vpn;
        interface ge-2/0/0.0;
        interface iw0.1;
        route-distinguisher 65000:5;
        vrf-target target:65000:2;
        protocols {
            l2vpn {
                no-control-word;
                encapsulation-type ethernet;
                site CE5 {
                    site-identifier 5;
                    interface ge-2/0/0.0 {
                        remote-site-id 3;
                    }
                }
                site l2-circuit {
                    site-identifier 6;
                    interface iw0.1 {
                        remote-site-id 3;
                    }
                }
            }
        }
    }
}
```

4. In addition to the `iw0` interface configuration, the Layer 2 interworking `l2iw` protocol must be configured. Without the `l2iw` protocol configuration, the Layer 2 interworking routes are not formed, regardless of whether any `iw` interfaces are present.

   On Router PE5, configure the `l2iw` protocol. To configure the protocol, include the `l2iw` statement at the `[edit protocols]` hierarchy level.

```
[edit]
protocols {
    l2iw;
}
```

Verification

Step-by-Step Procedure

Verifying the Layer 2 Circuit Connection on Router PE1.

1. On Router PE1, use the `show l2circuit connections` command to verify that the Layer 2 Circuit from Router PE1 to Router PE5 is `Up`.

```
user@PE1> show l2circuit connections
Layer-2 Circuit Connections:
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown
SP -- Static Pseudowire

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.5
    Interface                 Type  St     Time last up   # Up trans
    ge-1/0/0.0(vc 100)        rmt   Up     Jan  3 22:00:49 2010    1
      Remote PE: 192.0.2.5, Negotiated control-word: No
      Incoming label: 301328, Outgoing label: 300192
      Local interface: ge-1/0/0.0, Status: Up, Encapsulation: ETHERNET
```

2. On Router PE5, use the `show l2vpn connections` command to verify that the Layer 2 VPN connection is `Up` using the `iw0` peer interface of the Layer 2 circuit.

```
user@PE5> show l2vpn connections
Instance: L2VPN
  Local site: CE5 (5)
    connection-site           Type  St     Time last up     # Up trans
    l2-circuit (6)            loc   OR
    3                          rmt   Up
     Jan  3 22:51:12 2010          1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800258, Outgoing label: 800000
      Local interface: ge-2/0/0.0, Status: Up, Encapsulation: ETHERNET
  Local site: l2-circuit (6)
    connection-site            Type  St     Time last up    # Up trans
    CE5 (5)                    loc   OR
    3                          rmt   Up     Jan  3 22:56:38 2010    1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 800262, Outgoing label: 800001
      Local interface: iw0.1, Status: Up, Encapsulation: ETHERNET
```

Step-by-Step Procedure

Verifying that the Layer 2 Circuit is terminating into the Layer 2 VPN connection.

1. On Router PE5, use the `show l2circuit connections` command to verify that the Layer 2 circuit is `Up` using the `iw0` interface. This will be looped through the `iwo.1` interface to the Layer 2 VPN.

```
user@PE5> show l2circuit connections

Layer-2 Circuit Connections:
Neighbor: 192.0.2.1
    Interface            Type  St     Time last up  # Up trans
    iw0.0(vc 100)  rmt   Up Jan  3 21:59:07 2010   1
      Remote PE: 192.0.2.1, Negotiated control-word: No
      Incoming label: 300192, Outgoing label: 301328
```

2. On Router PE 5, use the `show route table mpls.0` command to verify the Layer 2 circuit and Layer 2 VPN routes. In the example below, the Layer 2 circuit is associated with LDP label `301328` and the Layer 2 VPN is associated with LDP label `800001`. Notice the two `iw0` interfaces that are used for the Layer 2 interworking route.

```
user@PE5>show route table mpls.0

mpls.0: 18 destinations, 20 routes (18 active, 2 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 5d 20:07:31, metric 1
```

```
                       Receive
1                    *[MPLS/0] 5d 20:07:31, metric 1
                       Receive
2                    *[MPLS/0] 5d 20:07:31, metric 1
                       Receive
299776               *[LDP/9] 2d 03:00:51, metric 1
300048               *[LDP/9] 2d 03:00:49, metric 1
                      > to 10.10.6.1 via xe-0/1/0.0, Pop
300048(S=0)          *[LDP/9] 2d 03:00:49, metric 1
                      > to 10.10.6.1 via xe-0/1/0.0, Pop
300192               *[L2IW/6] 19:11:05, metric2 1
                      > to 10.10.6.1 via xe-0/1/0.0, Swap 800001
                      [L2CKT/7] 20:08:36
                      > via iw0.0, Pop
800258               *[L2VPN/7] 19:16:31
                      > via ge-2/0/0.0, Pop       Offset: 4
800262             *[L2IW/6] 19:11:05, metric2 1                     > to 10.10.3.1 via xe-1/1/0.0,
Swap 301328                 [L2VPN/7] 19:11:05                       > via iw0.1, Pop       Offset:
4ge-2/0/0.0           *[L2VPN/7] 19:16:31, metric2 1
                      > to 10.10.6.1 via xe-0/1/0.0, Push 800000 Offset: -4
iw0.0                *[L2CKT/7] 20:08:36, metric2 1
                      > to 10.10.3.1 via xe-1/1/0.0, Push 301328
iw0.1                *[L2VPN/7] 19:11:05, metric2 1
                      > to 10.10.6.1 via xe-0/1/0.0, Push 800001 Offset: -4
```

RELATED DOCUMENTATION

No Link Title

Example: Interconnecting a Layer 2 Circuit with a Layer 2 Circuit

IN THIS SECTION

- Requirements | 1347
- Overview and Topology | 1347
- Configuration | 1349

This example provides a step-by-step procedure and commands for configuring and verifying a Layer 2 circuit to a Layer 2 circuit interconnection. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- 2 MX Series routers
- 2 M Series routers
- 1 T Series router
- 1 EX Series router

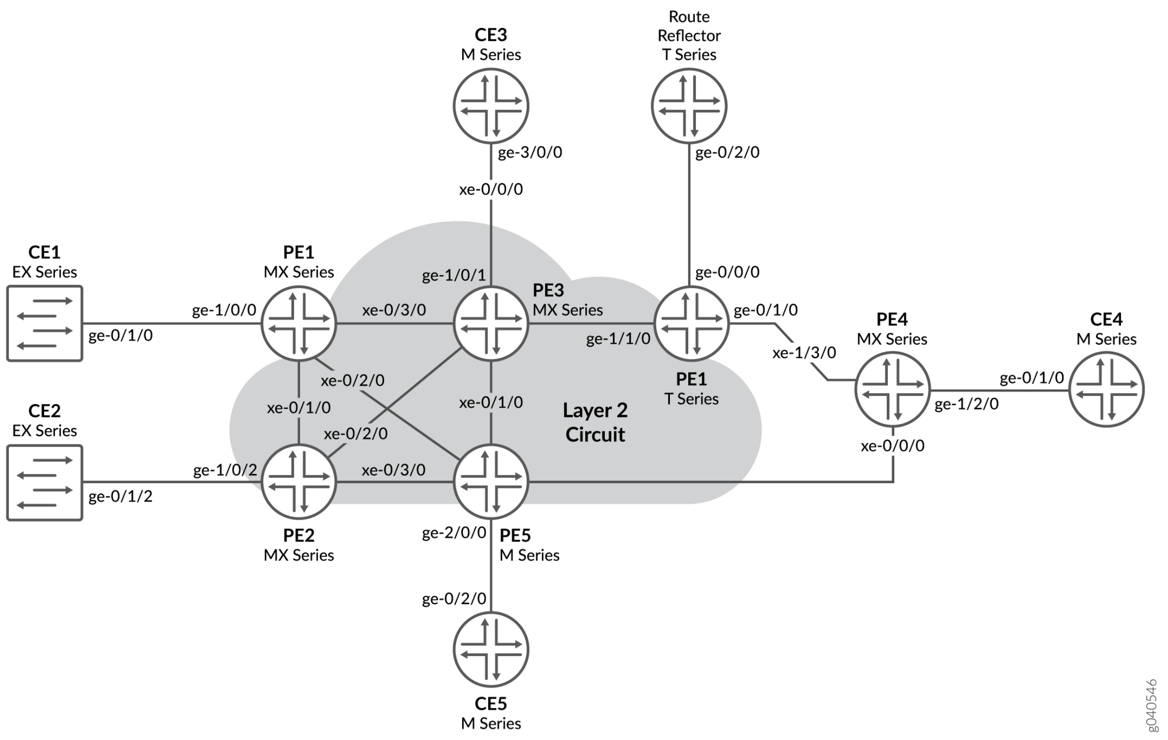
Overview and Topology

IN THIS SECTION

- Topology | **1349**

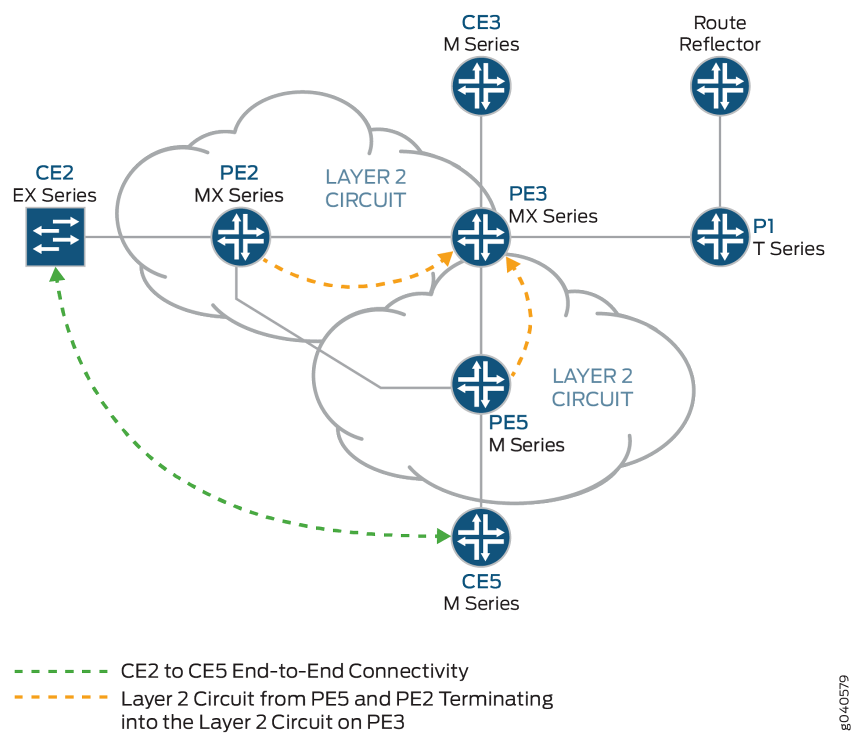
The physical topology of a Layer 2 circuit to Layer 2 circuit interconnection is shown in Figure 93 on page 1348

Figure 93: Physical Topology of a Layer 2 Circuit Terminating into a Layer 2 Circuit



The logical topology of a Layer 2 circuit to Layer 2 circuit interconnection is shown in Figure 94 on page 1349

Figure 94: Logical Topology of a Layer 2 Circuit Terminating into a Layer 2 Circuit



Topology

Configuration

IN THIS SECTION

- Configuring PE Router Customer-facing and Loopback Interfaces | **1350**
- Configuring Core-facing Interfaces | **1352**
- Configuring Protocols | **1354**
- Configuring the Layer 2 Circuits | **1355**
- Interconnecting the Layer 2 Circuits | **1357**
- Verifying the Layer 2 Circuit to Layer 2 Circuit Interconnection | **1358**
- Results | **1363**

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the `commit check` command.

In this example, the router being configured is identified using the following command prompts:

- `CE2` identifies the customer edge 2 (CE2) router
- `PE1` identifies the provider edge 1 (PE1) router
- `CE3` identifies the customer edge 3 (CE3) router
- `PE3` identifies the provider edge 3 (PE3) router
- `CE5` identifies the customer edge 5 (CE5) router
- `PE5` identifies the provider edge 5 (PE5) router

This example contains the following procedures:

Configuring PE Router Customer-facing and Loopback Interfaces

Step-by-Step Procedure

To begin building the interconnection, configure the interfaces on the PE routers. If your network contains provider (P) routers, configure the interfaces on the P routers also. This example shows the configuration for Router PE1 and Router PE5.

1. On Router PE1, configure the `ge-1/0/0` interface encapsulation. To configure the interface encapsulation, include the `encapsulation` statement and specify the `ethernet-ccc` option (vlan-ccc encapsulation is also supported). Configure the `ge-1/0/0.0` logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the `family` statement and specify the `ccc` option.

   ```
   [edit interfaces]
   ge-1/0/0 {
       encapsulation ethernet-ccc;
       unit 0 {
           family ccc;
       }
   }
   lo0 {
       unit 0 {
           family inet {
   ```

```
                address 192.0.2.1/24;
            }
        }
    }
```

2. On Router PE5, configure the `ge-2/0/0` interface encapsulation. To configure the interface encapsulation, include the `encapsulation` statement and specify the `ethernet-ccc` option. Configure the `ge-2/0/0.0` logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the `family` statement and specify the `ccc` option

```
[edit interfaces]
ge-2/0/0 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.5/24;
        }
    }
}
```

3. On Router PE3, configure the logical loopback interface. The loopback interface is used to establish the targeted LDP sessions to Routers PE1 and PE5.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
    }
}
```

Configuring Core-facing Interfaces

Step-by-Step Procedure

This procedure describes how to configure the core-facing interfaces on the PE routers. This example does not include all the core-facing interfaces shown in the physical topology illustration. Enable the `mpls` and `inet` address families on the core-facing interfaces.

1. On Router PE1, configure the `xe-0/3/0` interface. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify `10.10.1.1/30` as the interface address. Include the `family` statement and specify the `mpls` address family.

```
[edit interfaces]
xe-0/3/0 {
    unit 0 {
        family inet {
            address 10.10.1.1/30;
        }
        family mpls;
    }
}
```

2. On Router PE3, configure the core-facing interfaces. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify the IPv4 addresses shown in the example as the interface addresses. Include the `family` statement and specify the `mpls` address family. In the example, the `xe-0/0/0` interface is connected to the route reflector, the `xe-0/1/0` interface is connected to Router PE5, the `xe-0/2/0` interface is connected to Router PE2, and the `xe-0/3/0` interface is connected to Router PE1.

```
[edit interfaces]
xe-0/0/0 {
    unit 0 {
        family inet {
            address 10.10.20.2/30;
        }
        family mpls;
    }
}
xe-0/1/0 {
    unit 0 {
        family inet {
            address 10.10.6.1/30;
        }
        family mpls;
    }
}
xe-0/2/0 {
    unit 0 {
        family inet {
            address 10.10.5.2/30;
        }
        family mpls;
    }
}
xe-0/3/0 {
    unit 0 {
        family inet {
            address 10.10.1.2/30;
        }
        family mpls;
    }
}
```

3. On Router PE5, configure the `xe-0/1/0` interface. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify `10.10.6.2/30` as the interface address. Include the `family` statement and specify the `mpls` address family.

```
[edit interfaces]
xe-0/1/0 {
    unit 0 {
        family inet {
            address 10.10.6.2/30;
        }
        family mpls;
    }
}
```

Configuring Protocols

Step-by-Step Procedure

This procedure describes how to configure the protocols used in this example. If your network contains P routers, configure the protocols on the P routers also.

Configure all of the PE routers and P routers with OSPF as the IGP protocol. Enable MPLS and LDP protocols on all of the interfaces except `fxp.0`.

1. On Router PE1, enable OSPF as the IGP. Enable the MPLS and LDP protocols on all interfaces except `fxp.0`. LDP is used as the signaling protocol on Router PE1 for the Layer 2 circuit. The following configuration snippet shows the protocol configuration for Router PE1:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

2. Configure the PE and P routers with OSPF as the IGP. Enable the MPLS and LDP protocols on all interfaces except `fxp.0`. The following configuration snippet shows the protocol configuration for Router PE3:

```
[edit]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

Configuring the Layer 2 Circuits

Step-by-Step Procedure

This procedure describes how to configure the Layer 2 circuits.

NOTE: In this example the `ignore-mtu-mismatch` statement is required for the circuit to come up.

1. On Router PE1, configure the Layer 2 circuit. Include the `l2circuit` statement. Include the `neighbor` statement and specify the loopback IPv4 address of Router PE3 as the neighbor. Include the

interface statement and specify ge-1/0/0.0 as the logical interface that is participating in the Layer 2 circuit. Include the virtual-circuit-id statement and specify 100 as the identifier. Include the ignore-mtu-mismatch statement to allow a Layer 2 circuit to be established even though the maximum transmission unit (MTU) configured on the local PE router does not match the MTU configured on the remote PE router.

```
[edit]
protocols {
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-1/0/0.0 {
                virtual-circuit-id 100;
                ignore-mtu-mismatch;
            }
        }
    }
}
```

2. On Router PE5, configure the Layer 2 circuit. Include the l2circuit statement. Include the neighbor statement and specify the loopback IPv4 address of Router PE3 as the neighbor. Include the interface statement and specify ge-2/0/0.0 as the logical interface that is participating in the Layer 2 circuit. Include the virtual-circuit-id statement and specify 200 as the identifier. Include the ignore-mtu-mismatch statement to allow a Layer 2 circuit to be established even though the MTU configured on the local PE router does not match the MTU configured on the remote PE router.

```
[edit]
protocols {
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-2/0/0.0 {
                virtual-circuit-id 200;
                ignore-mtu-mismatch;
            }
        }
    }
}
```

3. On Router PE3, configure the Layer 2 circuit to Router PE1. Include the l2circuit statement. Include the neighbor statement and specify the loopback IPv4 address of Router PE1 as the neighbor. Include the interface statement and specify iw0.0 as the logical interworking interface that is participating in

the Layer 2 circuit. Include the `virtual-circuit-id` statement and specify `100` as the identifier. Include the `ignore-mtu-mismatch` statement to allow a Layer 2 circuit to be established even though the MTU configured on the local PE router does not match the MTU configured on the remote PE router.

On Router PE3, configure the Layer 2 circuit to Router PE5. Include the `l2circuit` statement. Include the `neighbor` statement and specify the loopback IPv4 address of Router PE5 as the neighbor. Include the interface statement and specify `iw0.1` as the logical interworking interface that is participating in the Layer 2 circuit. Include the `virtual-circuit-id` statement and specify `200` as the identifier. Include the `ignore-mtu-mismatch` statement.

```
[edit protocols]
l2circuit {
    neighbor 192.0.2.1 {
        interface iw0.0 {
            virtual-circuit-id 100;
            ignore-mtu-mismatch;
        }
    }
    neighbor 192.0.2.5 {
        interface iw0.1 {
            virtual-circuit-id 200;
            ignore-mtu-mismatch;
        }
    }
}
```

Interconnecting the Layer 2 Circuits

Step-by-Step Procedure

Router PE3 is the router that is *stitching* the Layer 2 circuits together using the interworking interface. The configuration of the peer unit interfaces is what makes the interconnection.

1. On Router PE3, configure the `iw0.0` interface. Include the `encapsulation` statement and specify the `ethernet-ccc` option. Include the `peer-unit` statement and specify the logical interface unit `1` as the peer tunnel interface.

On Router PE3, configure the `iw0.1` interface. Include the `encapsulation` statement and specify the `ethernet-ccc` option. Include the `peer-unit` statement and specify the logical interface unit `0` as the peer tunnel interface.

```
[edit interfaces]
iw0 {
    unit 0 {
        encapsulation ethernet-ccc;
        peer-unit 1;
    }
    unit 1 {
        encapsulation ethernet-ccc;
        peer-unit 0;
    }
}
```

2. On Router PE3, configure the Layer 2 interworking `l2iw` protocol. To configure the Layer 2 interworking protocol, include the `l2iw` statement at the `[edit protocols]` hierarchy level.

```
[edit]
protocols {
    l2iw;
}
```

3. On each router, commit the configuration.

```
user@host> commit check
configuration check succeeds
user@host> commit
```

Verifying the Layer 2 Circuit to Layer 2 Circuit Interconnection

Step-by-Step Procedure

Verify that the Layer 2 circuit connection on Router PE1 is up, the LDP neighbors are correct, and the MPLS label operations are correct.

1. On Router PE1, use the `show l2circuit connections` command to verify that the Layer 2 circuit from Router PE1 to Router PE3 is `Up`.

```
user@PE1> show l2circuit connections
Layer-2 Circuit Connections:
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown
SP -- Static Pseudowire

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.3
    Interface                 Type  St     Time last up   # Up trans
    ge-1/0/0.0(vc 100)        rmt   Up     Jan  5 22:00:49 2010    1
      Remote PE: 192.0.2.3, Negotiated control-word: Yes (Null)
      Incoming label: 301328, Outgoing label: 314736
      Local interface: ge-1/0/0.0, Status: Up, Encapsulation: ETHERNET
```

2. On Router PE1, use the `show ldp neighbor` command to verify that the IPv4 address of Router PE3 is shown as the LDP neighbor.

```
user@PE1> show ldp neighbor

Address            Interface          Label space ID         Hold time
192.0.2.3            lo0.0              192.0.2.3:0                41
```

3. On Router PE 1, use the `show route table mpls.0` command to verify the Layer 2 circuit is using the LDP label to Router PE3 in both directions (Push and Pop). In the example below, the Layer 2 circuit is associated with LDP label `301328`.

```
user@PE1> show route table mpls.0

mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w1d 08:25:39, metric 1
                      Receive
1                  *[MPLS/0] 1w1d 08:25:39, metric 1
                      Receive
2                  *[MPLS/0] 1w1d 08:25:39, metric 1
                      Receive
300432             *[LDP/9] 3d 01:13:57, metric 1
                    > to 10.10.2.2 via xe-0/1/0.0, Pop
300432(S=0)        *[LDP/9] 3d 01:13:57, metric 1
                    > to 10.10.2.2 via xe-0/1/0.0, Pop
300768             *[LDP/9] 3d 01:13:57, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Pop
300768(S=0)        *[LDP/9] 3d 01:13:57, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Pop
300912             *[LDP/9] 3d 01:13:57, metric 1
                    > to 10.10.3.2 via xe-0/2/0.0, Swap 299856
301264             *[LDP/9] 3d 01:13:53, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Swap 308224
301312             *[LDP/9] 3d 01:13:56, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Pop
301312(S=0)        *[LDP/9] 3d 01:13:56, metric 1
                    > to 10.10.1.2 via xe-0/3/0.0, Pop
301328             *[L2CKT/7] 02:33:26                    > via ge-1/0/0.0, Pop       Offset: 4
ge-1/0/0.0         *[L2CKT/7] 02:33:26, metric2 1                    > to 10.10.1.2 via xe-0/3/0.0, Push
314736 Offset: -4
```

4. On Router PE3, use the `show l2circuit connections` command to verify that the Layer 2 circuit from Router PE3 to Router PE5 is `Up`, that the Layer 2 circuit from Router PE3 to Router PE1 is `Up`, that the

connections to Router PE1 and Router PE5 use the iw0 interface, and that the status for both local iw0 interfaces is `Up`.

```
user@PE3> show l2circuit connections
Layer-2 Circuit Connections:
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown
SP -- Static Pseudowire

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.1
    Interface                 Type  St     Time last up          # Up trans
    iw0.0(vc 100)             rmt   Up     Jan  5 13:50:14 2010           1
      Remote PE: 192.0.2.1, Negotiated control-word: Yes (Null)
      Incoming label: 314736, Outgoing label: 301328
      Local interface: iw0.0, Status: Up, Encapsulation: ETHERNET
Neighbor: 192.0.2.5
    Interface                 Type  St     Time last up          # Up trans
    iw0.1(vc 200)             rmt   Up     Jan  5 13:49:58 2010           1
      Remote PE: 192.0.2.5, Negotiated control-word: Yes (Null)
      Incoming label: 314752, Outgoing label: 300208
      Local interface: iw0.1, Status: Up, Encapsulation: ETHERNET
```

5. On Router PE3, use the `show ldp neighbor` command to verify that the correct IPv4 addresses are shown as the LDP neighbor.

```
user@PE3> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.0.2.1            lo0.0              192.0.2.1:0              44
192.0.2.2            lo0.0              192.0.2.2:0              42
```

```
192.0.2.4            lo0.0              192.0.2.4:0               31
192.0.2.5            lo0.0              192.0.2.5:0               44
```

6. On Router PE3, use the `show route table mpls.0` command to verify that the `mpls.0` routing table is populated with the Layer 2 interworking routes. Notice that in this example, the router is swapping label `314736` received from Router PE1 on the `iw0.0` to label `301328`.

```
user@PE3> show route table mpls.0
mpls.0: 16 destinations, 18 routes (16 active, 2 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w1d 08:28:24, metric 1
                      Receive
1                  *[MPLS/0] 1w1d 08:28:24, metric 1
                      Receive
2                  *[MPLS/0] 1w1d 08:28:24, metric 1
                      Receive
308160             *[LDP/9] 3d 01:16:55, metric 1
                    > to 10.10.1.1 via xe-0/3/0.0, Pop
308160(S=0)        *[LDP/9] 3d 01:16:55, metric 1
                    > to 10.10.1.1 via xe-0/3/0.0, Pop
308176             *[LDP/9] 3d 01:16:54, metric 1
                    > to 10.10.6.2 via xe-0/1/0.0, Pop
308176(S=0)        *[LDP/9] 3d 01:16:54, metric 1
                    > to 10.10.6.2 via xe-0/1/0.0, Pop
308192             *[LDP/9] 00:21:40, metric 1
                    > to 10.10.20.1 via xe-0/0/0.0, Swap 601649
                      to 10.10.6.2 via xe-0/1/0.0, Swap 299856
308208             *[LDP/9] 3d 01:16:54, metric 1
                    > to 10.10.5.1 via xe-0/2/0.0, Pop
308208(S=0)        *[LDP/9] 3d 01:16:54, metric 1
                    > to 10.10.5.1 via xe-0/2/0.0, Pop
308224             *[LDP/9] 3d 01:16:52, metric 1
                    > to 10.10.20.1 via xe-0/0/0.0, Pop
308224(S=0)        *[LDP/9] 3d 01:16:52, metric 1
                    > to 10.10.20.1 via xe-0/0/0.0, Pop
314736             *[L2IW/6] 02:35:31, metric2 1
                    > to 10.10.6.2 via xe-0/1/0.0, Swap 300208
                    [L2CKT/7] 02:35:31
                    > via iw0.0, Pop       Offset: 4
314752             *[L2IW/6] 02:35:31, metric2 1
                    > to 10.10.1.1 via xe-0/3/0.0, Swap 301328
                    [L2CKT/7] 02:35:47
                    > via iw0.1, Pop       Offset: 4
iw0.0              *[L2CKT/7] 02:35:31, metric2 1
                    > to 10.10.1.1 via xe-0/3/0.0, Push 301328 Offset: -4
iw0.1              *[L2CKT/7] 02:35:47, metric2 1
                    > to 10.10.6.2 via xe-0/1/0.0, Push 300208 Offset: -4
```

7. Verify that Router CE1 can send traffic to and receive traffic from Router CE5 across the interconnection, using the `ping` command.

```
user@CE1>ping 198.51.100.11
PING 198.51.100.11 (198.51.100.11): 56 data bytes
64 bytes from 198.51.100.11: icmp_seq=1 ttl=64 time=22.425 ms
64 bytes from 198.51.100.11: icmp_seq=2 ttl=64 time=1.299 ms
64 bytes from 198.51.100.11: icmp_seq=3 ttl=64 time=1.032 ms
64 bytes from 198.51.100.11: icmp_seq=4 ttl=64 time=1.029 ms
```

8. Verify that Router CE5 can send traffic to and receive traffic from Router CE1 across the interconnection, using the `ping` command.

```
user@CE5>ping 198.51.100.1
PING 198.51.100.1 (198.51.100.1): 56 data bytes
64 bytes from 198.51.100.1: icmp_seq=0 ttl=64 time=1.077 ms
64 bytes from 198.51.100.1: icmp_seq=1 ttl=64 time=0.957 ms
64 bytes from 198.51.100.1: icmp_seq=2 ttl=64 time=1.057 ms 1.017 ms
```

Results

The configuration and verification of this example has been completed. The following section is for your reference.

The relevant sample configuration for Router PE1 follows.

Router PE1

```
[edit]
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.1/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.3..1/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.1/30;
            }
            family mpls;
        }
    }
    ge-1/0/0 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.1/24;
            }
        }
    }
}
forwarding-options {
    hash-key {
        family inet {
            layer-3;
            layer-4;
        }
        family mpls {
            label-1;
            label-2;
        }
    }
}
routing-options {
    static {
        route 172.16.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-1/0/0.0 {
                virtual-circuit-id 100;
                ignore-mtu-mismatch;
            }
        }
    }
}
```

The relevant sample configuration for Router PE3 follows.

Router PE3

```
[edit]
interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                address 10.10.20.2/30;
            }
            family mpls;
        }
    }
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.6.1/30;
            }
            family mpls;
        }
    }
    xe-0/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.2/30;
            }
            family mpls;
        }
    }
    xe-0/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.2/30;
            }
            family mpls;
        }
    }
    ge-1/0/1 {
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
    iw0 {
        unit 0 {
            encapsulation ethernet-ccc;
            peer-unit 1;
        }
        unit 1 {
            encapsulation ethernet-ccc;
            peer-unit 0;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.0.2.3/24;
            }
        }
    }
}
routing-options {
    static {
        route 172.16.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
    l2iw;
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    l2circuit {
        neighbor 192.0.2.1 {
            interface iw0.0 {
                virtual-circuit-id 100;
                ignore-mtu-mismatch;
            }
        }
        neighbor 192.0.2.5 {
            interface iw0.1 {
                virtual-circuit-id 200;
                ignore-mtu-mismatch;
            }
        }
    }
}
```

RELATED DOCUMENTATION

No Link Title

Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN

MPLS-based Layer 2 services are growing in demand among enterprise and service providers. This creates new challenges related to interoperability between Layer 2 and Layer 3 services for service providers who want to provide end-to-end value-added services. There are various reasons to stitch different Layer 2 services to one another and to Layer 3 services. For example, to expand the service offerings and to expand geographically. The Junos OS has various features to address the needs of the service provider.

You can enable pseudowire services and configure a pseduowire service interface as an access point for interconnecting layer 2 circuits to layer 3 VPNs. For more information, see Pseudowire Subscriber Logical Interfaces Overview.

Interconnecting a Layer 2 Circuit with a Layer 3 VPN provides the following benefits:

- Interconnecting a Layer 2 Circuit with a Layer 3 VPN enables the sharing of a service provider's core network infrastructure between IP and Layer 2 circuit services, reducing the cost of providing those services. A Layer 2 MPLS circuit allows service providers to create a Layer 2 circuit service over an existing IP and MPLS backbone.
- Service providers do not have to invest in separate Layer 2 equipment to provide Layer 2 circuit service. A service provider can configure a provider edge router to run any Layer 3 protocol in addition to the Layer 2 protocols. Customers who prefer to maintain control over most of the administration of their own networks want Layer 2 circuit connections with their service provider instead of a Layer 3 VPN connection.

Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN

IN THIS SECTION

- Requirements | **1369**
- Overview and Topology | **1370**
- Configuration | **1371**
- Verifying the Layer 2 Circuit to Layer 3 VPN Interconnection | **1385**

This example provides a step-by-step procedure and commands for configuring and verifying a Layer 2 circuit to Layer 3 VPN interconnection. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- 3 MX Series 5G Universal Routing Platforms
- 1 M Series Multiservice Edge Router
- 1 T Series Core Router
- 1 EX Series Ethernet Switch

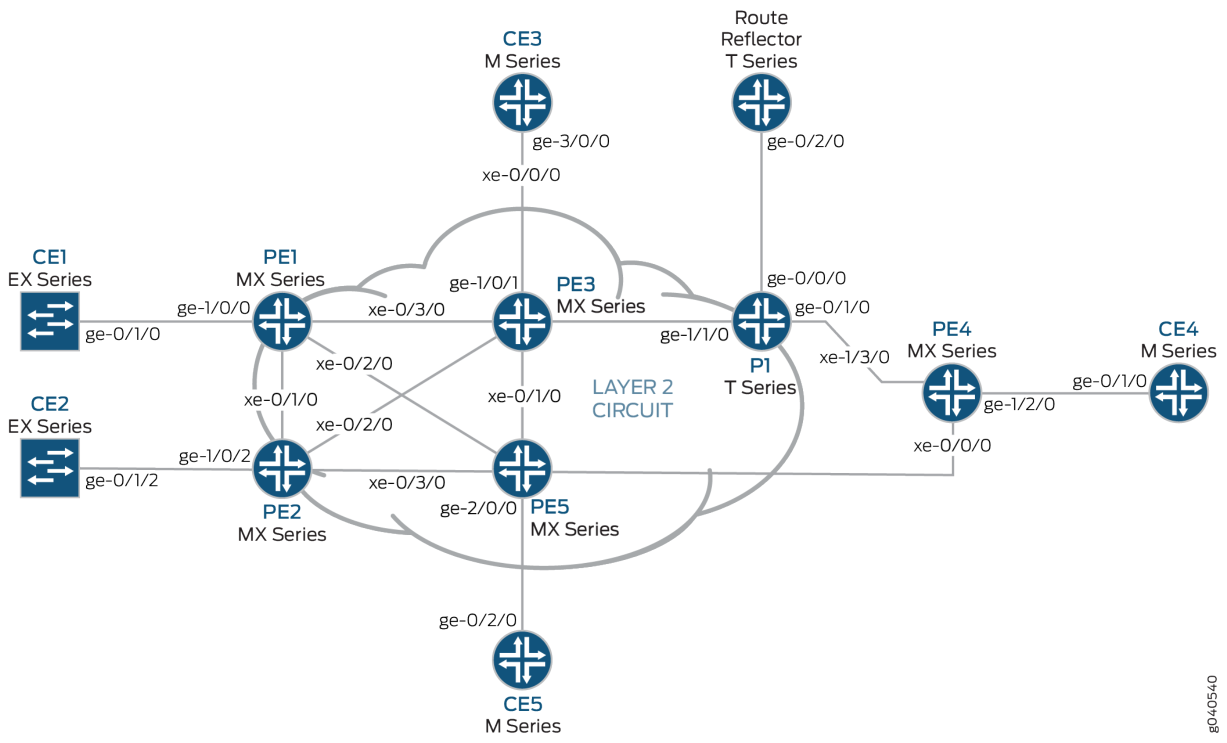
Overview and Topology

IN THIS SECTION

- Topology | **1371**

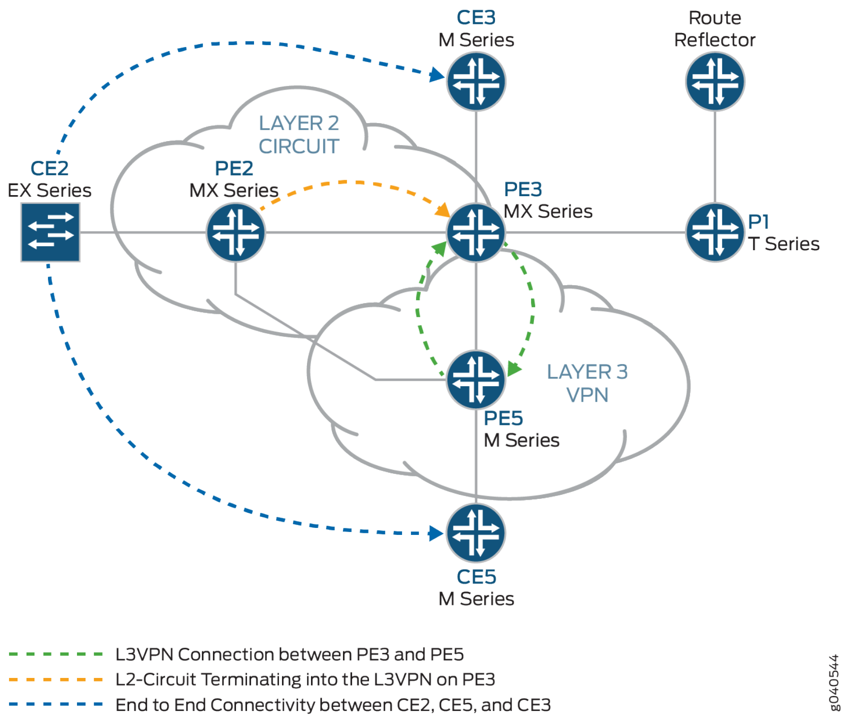
The physical topology of a Layer 2 circuit to Layer 3 VPN interconnection is shown in Figure 95 on page 1370.

Figure 95: Physical Topology of a Layer 2 Circuit to Layer 3 VPN Interconnection



The logical topology of a Layer 2 circuit to Layer 3 VPN interconnection is shown in Figure 96 on page 1371.

Figure 96: Logical Topology of a Layer 2 Circuit to Layer 3 VPN Interconnection


CE3
M Series
Route
Reflector
LAYER 2
CIRCUIT
CE2
EX Series
PE2
MX Series
PE3
MX Series
P1
T Series
LAYER 3
VPN
PE5
M Series
CE5
M Series
L3VPN Connection between PE3 and PE5
L2-Circuit Terminating into the L3VPN on PE3
End to End Connectivity between CE2, CE5, and CE3
g040544

Topology

Configuration

IN THIS SECTION

- Configuring PE Router Customer-facing and Loopback Interfaces | **1372**
- Configuring Core-facing Interfaces | **1374**
- Configuring Protocols | **1376**
- Configuring Routing Instances and Layer 2 Circuits | **1380**
- Configuring the Route Reflector | **1382**
- Interconnecting the Layer 2 Circuit with the Layer 3 VPN | **1384**

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the `commit check` command.

In this example, the router being configured is identified using the following command prompts:

- `CE2` identifies the customer edge 2 (CE2) router
- `PE1` identifies the provider edge 1 (PE1) router
- `CE3` identifies the customer edge 3 (CE3) router
- `PE3` identifies the provider edge 3 (PE3) router
- `CE5` identifies the customer edge 5 (CE5) router
- `PE5` identifies the provider edge 5 (PE5) router

This example contains the following procedures:

Configuring PE Router Customer-facing and Loopback Interfaces

Step-by-Step Procedure

To begin building the interconnection, configure the interfaces on the PE routers. If your network contains provider (P) routers, configure the interfaces on the P routers also. This example shows the configuration for Router PE2, Router PE3, and Router PE5.

1. On Router PE2, configure the `ge-1/0/2` interface encapsulation. To configure the interface encapsulation, include the `encapsulation` statement and specify the `ethernet-ccc` option (`vlan-ccc` encapsulation is also supported). Configure the `ge-1/0/2.0` logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the `family` statement and specify the `ccc` option. The encapsulation should be configured the same way for all routers in the Layer 2 circuit domain.

```
[edit interfaces]
ge-1/0/2 {
    encapsulation ethernet-ccc;
    unit 0 {
        family ccc;
    }
}
```

2. On Router PE2, configure the `lo0.0` interface. Include the `family` statement and specify the `inet` option. Include the `address` statement and specify `192.0.2.2/24` as the loopback IPv4 address.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.2/24;
        }
    }
}
```

3. On Router PE3, configure the `ge-1/0/1` interface. Include the `family` statement and specify the `inet` option. Include the `address` statement and specify `198.51.100.1/24` as the interface address for this device.

```
[edit interfaces]
ge-1/0/1 {
    unit 0 {
        family inet {
            address 198.51.100.1/24;
        }
    }
}
```

4. On Router PE3, configure the `lo0.0` loopback interface. Include the `family` statement and specify the `inet` option. Include the `address` statement and specify `192.0.2.3/24` as the loopback IPv4 address for this router.

```
[edit interfaces]
lo0 {
    unit 0 {
        family inet {
            address 192.0.2.3/24;
        }
    }
}
```

5. On Router PE5, configure the `ge-2/0/0` interface. Include the `family` statement and specify the `inet` option. Include the `address` statement and specify `198.51.100.8/24` as the interface address.

   ```
   [edit interfaces]
   ge-2/0/0 {
       unit 0 {
           family inet {
               address 198.51.100.8/24;
           }
       }
   }
   ```

6. On Router PE5, configure the `lo0.0` interface. Include the `family` statement and specify the `inet` option. Include the `address` statement and specify `192.0.2.5/24` as the loopback IPv4 address for this router.

   ```
   [edit interfaces]
   lo0 {
       unit 0 {
           family inet {
               address 192.0.2.5/24;
           }
       }
   }
   ```

Configuring Core-facing Interfaces

Step-by-Step Procedure

This procedure describes how to configure the core-facing interfaces on the PE routers. This example does not include all the core-facing interfaces shown in the physical topology illustration. Enable the `mpls` and `inet` address families on the core-facing interfaces.

1. On Router PE2, configure the `xe-0/2/0` interface. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify `10.10.5.1/30` as the interface address. Include the `family` statement and specify the `mpls` address family.

   ```
   [edit interfaces]
   xe-0/2/0 {
       unit 0 {
           family inet {
   ```

```
            address 10.10.5.1/30;
        }
        family mpls;
    }
}
```

2. On Router PE3, configure the core-facing interfaces. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify the IPv4 addresses shown in the example as the interface addresses. Include the `family` statement and specify the `mpls` address family. In the example, the `xe-2/1/0` interface is connected to Router PE5, and the `xe-2/2/0` interface is connected to Router PE2.

```
[edit interfaces]
xe-2/0/0 {
    unit 0 {
        family inet {
            address 10.10.20.2/30;
        }
        family mpls;
    }
}
xe-2/1/0 {
    unit 0 {
        family inet {
            address 10.10.6.1/30;
        }
        family mpls;
    }
}
xe-2/2/0 {
    unit 0 {
        family inet {
            address 10.10.5.2/30;
        }
        family mpls;
    }
}
xe-2/3/0 {
    unit 0 {
        family inet {
            address 10.10.1.2/30;
        }
        family mpls;
    }
}
```

3. On Router PE5, configure the `xe-0/1/0` interface. Include the `family` statement and specify the `inet` address family. Include the `address` statement and specify `10.10.6.2/30` as the interface address. Include the `family` statement and specify the `mpls` address family.

```
[edit interfaces]
xe-0/1/0 {
    unit 0 {
        family inet {
            address 10.10.6.2/30;
        }
        family mpls;
    }
}
```

Configuring Protocols

Step-by-Step Procedure

This procedure describes how to configure the protocols used in this example. If your network contains P routers, configure the interfaces on the P routers also.

1. On Router PE3, enable OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except `fxp.0`. LDP is used as the signaling protocol for the Layer 2 circuit to Router PE2 . The following configuration snippet shows the protocol configuration for Router PE3:

```
[edit]
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE5 {
            to 192.0.2.5;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.3;
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

2. On Router PE2, configure the MPLS, OSPF, and LDP protocols.

```
[edit ]
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

3. On Router PE5, enable OSPF as the IGP. Enable the MPLS, RSVP, and BGP protocols on all interfaces except `fxp.0`. Enable core-facing interfaces with the `mpls` and `inet` address families.

```
[edit]
protocols {
    rsvp {
        interface all {
            link-protection;
        }
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-RR {
            to 192.0.2.7;
        }
        label-switched-path to-PE2 {
            to 192.0.2.2;
        }
        label-switched-path to-PE3 {
            to 192.0.2.3;
        }
        label-switched-path to-PE4 {
            to 192.0.2.4;
        }
        label-switched-path to-PE1 {
            to 192.0.2.1;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group to-rr {
            type internal;
            local-address 192.0.2.5;
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            neighbor 192.0.2.7;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
```

Configuring Routing Instances and Layer 2 Circuits

Step-by-Step Procedure

This procedure describes how to configure the Layer 2 circuit and the Layer 3 VPN.

1. On Router PE2, configure the Layer 2 circuit. Include the `l2circuit` statement. Include the `neighbor` statement and specify the loopback IPv4 address of Router PE3 as the neighbor. Include the interface statement and specify `ge-1/0/2.0` as the logical interface that is participating in the Layer 2 circuit. Include the `virtual-circuit-id` statement and specify `100` as the identifier. Include the `no-control-word` statement for equipment that does not support the control word.

```
[edit ]
protocols {
    l2circuit {
        neighbor 192.0.2.3 {
            interface ge-1/0/2.0 {
                virtual-circuit-id 100;
                no-control-word;
            }
        }
    }
}
```

2. On Router PE3, configure the Layer 2 circuit to Router PE2. Include the `l2circuit` statement. Include the `neighbor` statement and specify the loopback IPv4 address of Router PE2 as the neighbor. Include the interface statement and specify `lt-1/1/10.0` as the logical tunnel interface that is participating in the Layer 2 circuit. Include the `virtual-circuit-id` statement and specify `100` as the identifier. Include the `no-control-word` statement.

```
[edit ]
protocols {
    l2circuit {
        neighbor 192.0.2.2 {
            interface lt-1/1/10.0 {
                virtual-circuit-id 100;
                no-control-word;
            }
        }
    }
}
```

3. On Router PE3, configure the Layer 3 VPN (`L3VPN`) routing instance to Router PE5 at the `[edit routing-instances]` hierarchy level. Also configure the BGP peer group at the `[edit routing-instances L3VPN protocols]` hierarchy level.

```
[edit ]
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-1/0/1.0;
        interface lt-1/1/10.1;
        route-distinguisher 65000:33;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                export direct;
                group ce3 {
                    neighbor 198.51.100.6{
                        peer-as 100;
                    }
                }
            }
        }
    }
}
```

4. On Router PE5, configure the Layer 3 VPN routing instance (`L3VPN`) at the `[edit routing-instances]` hierarchy level. Also configure the BGP peer group at the `[edit routing-instances L3VPN protocols]` hierarchy level.

```
[edit ]
routing-instances {
    L3VPN {
        instance-type vrf;
        interface ge-2/0/0.0;
        route-distinguisher 65000:5;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
            bgp {
                group ce5 {
                    neighbor 198.51.100.10 {
                        peer-as 200;
                    }
                }
            }
        }
    }
}
```

Configuring the Route Reflector

Step-by-Step Procedure

Although a route reflector is not required to interconnect a Layer 2 circuit with a Layer 3 VPN, this examples uses a route reflector. This procedure shows the relevant portion of the route reflector configuration.

1. Configure the route reflector with RSVP, MPLS, BGP and OSPF. The route reflector is a BGP peer with the PE routers. Notice that the BGP peer group configuration includes the `family` statement and specifies the `inet-vpn` option The `inet-vpn` option enables BGP to advertise network layer reachability information (NLRI) for the Layer 3 VPN routes. The configuration also includes the `family` statement and specifies the `l2vpn` option. The `l2vpn` option enables BGP to advertise NLRI for the Layer 2 circuit. Layer 2 circuits use the same internal BGP infrastructure as Layer 2 VPNs.

```
[edit ]
protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    mpls {
        label-switched-path to-pe3 {
            to 192.0.2.3;
        }
        label-switched-path to-pe5 {
            to 192.0.2.5;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    bgp {
        group RR {
            type internal;
            local-address 192.0.2.7;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family l2vpn {
                signaling;
            }
            cluster 192.0.2.7;
            neighbor 192.0.2.1;
            neighbor 192.0.2.2;
            neighbor 192.0.2.4;
            neighbor 192.0.2.5;
            neighbor 192.0.2.3;
        }
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }
}
```

Interconnecting the Layer 2 Circuit with the Layer 3 VPN

Step-by-Step Procedure

Before you can configure the logical tunnel interface in an MX Series router, you must create the tunnel services interface to be used for tunnel services.

1. Create the tunnel service interface on Router PE3. Include the `bandwidth` statement at the `[edit chassis fpc *slot-number* pic *slot-number* tunnel-services]` hierarchy level and specify the amount of bandwidth to reserve for tunnel services in gigabits per second.

```
[edit chassis]
fpc 1 {
    pic 1 {
        tunnel-services {
            bandwidth 1g;
        }
    }
}
```

2. On Router PE3, configure the `lt-1/1/10` logical tunnel interface unit 0.

   Router PE3 is the router that is *stitching* the Layer 2 circuit to the Layer 3 VPN using the logical tunnel interface. The configuration of the peer unit interfaces is what makes the interconnection.

   Include the `encapsulation` statement and specify the `ethernet-ccc` option. Include the `peer-unit` statement and specify the logical interface unit `1` as the peer tunnel interface. Include the `family` statement and specify the `ccc` option.

   Configure the `lt-1/1/10` logical interface unit `1` with `ethernet` encapsulation. Include the `peer-unit` statement and specify the logical interface unit `0` as the peer tunnel interface. Include the `family` statement and specify the `inet` option. Also include the `address` statement and specify `198.51.100.11/24` as the IPv4 address of the interface.

   NOTE: The peering logical interfaces must belong to the same logical tunnel interface derived from the Tunnel Services PIC.

```
[edit interfaces]
lt-1/1/10 {
    unit 0 {
        encapsulation ethernet-ccc;
        peer-unit 1;
        family ccc;
    }
    unit 1 {
        encapsulation ethernet;
        peer-unit 0;
        family inet {
            address 198.51.100.11/24;
        }
    }
}
```

3. On each router, commit the configuration.

```
user@host> commit check
configuration check succeeds
user@host> commit
```

Verifying the Layer 2 Circuit to Layer 3 VPN Interconnection

IN THIS SECTION

- Verifying That the Layer 2 Circuit Connection to Router PE3 is Up | **1386**
- Verifying LDP Neighbors and Targeted LDP LSPs on Router PE2 | **1387**
- Verifying the Layer 2 Circuit Routes on Router PE2 | **1387**
- Verifying That the Layer 2 Circuit Connection to Router PE2 is Up | **1388**
- Verifying LDP Neighbors and Targeted LDP LSPs on Router PE3 | **1389**
- Verifying a BGP Peer Session with the Route Reflector on Router PE3 | **1390**
- Verifying the Layer 3 VPN Routes on Router PE3 | **1391**
- Verifying the Layer 2 Circuit Routes on Router PE3 | **1392**
- Verifying the MPLS Routes on Router PE3 | **1392**
- Verifying Traffic Flow Between Router CE2 and Router CE3 | **1394**
- Verifying Traffic Flow Between Router CE2 and Router CE5 | **1395**

To verify that the interconnection is working properly, perform these tasks:

Verifying That the Layer 2 Circuit Connection to Router PE3 is Up

Purpose

To verify that the Layer 2 circuit connection from Router PE2 to Router PE3 is `Up`. To also document the incoming and outgoing LDP labels and the circuit ID used by this Layer 2 circuit connection.

Action

Verify that the Layer 2 circuit connection is up, using the `show l2circuit connections` command.

```
user@PE2> show l2circuit connections
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.3
    Interface                 Type  St     Time last up          # Up trans
    ge-1/0/2.0(vc 100)        rmt   Up     Jan  7 02:14:13 2010           1
      Remote PE: 192.0.2.3, Negotiated control-word: No
      Incoming label: 301488, Outgoing label: 315264
      Negotiated PW status TLV: No
      Local interface: ge-1/0/2.0, Status: Up, Encapsulation: ETHERNET
```

Meaning

The output shows that the Layer 2 circuit connection from Router PE2 to Router PE3 is `Up` and the connection is using the `ge-1/0/2.0` interface. Note that the outgoing label is `315264` and the incoming label is `301488`, the virtual circuit (VC) identifier is `100` and the encapsulation is `ETHERNET`.

Verifying LDP Neighbors and Targeted LDP LSPs on Router PE2

Purpose

To verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors.

Action

Verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors, using the `show ldp neighbor` command.

```
user@PE2> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.0.2.3            lo0.0              192.0.2.3:0                38
```

Meaning

The output shows that Router PE2 has an LDP neighbor with the IPv4 address of `192.0.2.3`. Address 192.0.2.3 is the lo0.0 interface address of Router PE3. Notice that Router PE2 uses the local `lo0.0` interface for the LSP.

Verifying that the routers are LDP neighbors also verifies that the targeted LSP is established.

Verifying the Layer 2 Circuit Routes on Router PE2

Purpose

To verify that Router PE2 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE3.

Action

Verify that Router PE2 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE3, using the `show route table mpls.0` command.

```
user@PE2> show route table mpls.0
mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
0                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
300560             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
300560(S=0)        *[LDP/9] 16:12:23, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
301008             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Swap 299856
301488             *[L2CKT/7] 11:07:28
                    > via ge-1/0/2.0, Pop
301536             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301536(S=0)        *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301712             *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Swap 315184
301728             *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
301728(S=0)        *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
ge-1/0/2.0         *[L2CKT/7] 11:07:28, metric2 1
                    > to 10.10.5.2 via xe-0/2/0.0, Push 315264
```

Meaning

The output shows that Router PE2 pushes the `315264` outgoing label on the `L2CKT` route going out interface `ge-1/0/2.0`. The output also shows that Router PE2 pops the `301488` incoming label on the `L2CKT` coming from interface `ge-1/0/2.0`

Verifying That the Layer 2 Circuit Connection to Router PE2 is Up

Purpose

To verify that the Layer 2 circuit connection from Router PE3 to Router PE2 is `Up`, To also document the incoming and outgoing LDP labels and the circuit ID used by this Layer 2 circuit connection.

Action

Verify that the Layer 2 circuit connection is up, using the `show l2circuit connections` command.

```
user@PE3> show l2circuit connections
Layer-2 Circuit Connections:
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 192.0.2.2
    Interface                 Type  St     Time last up          # Up trans
    lt-1/1/10.0(vc 100)       rmt   Up     Jan  7 02:15:03 2010           1
      Remote PE: 192.0.2.2, Negotiated control-word: No
      Incoming label: 315264, Outgoing label: 301488
      Local interface: lt-1/1/10.0, Status: Up, Encapsulation: ETHERNET
```

Meaning

The output shows that the Layer 2 circuit connection from Router PE3 to Router PE2 is `Up` and the connection is using the logical tunnel (`lt`) interface. Note that the incoming label is `315264` and the outgoing label is `301488`, the virtual circuit (VC) identifier is `100`, and that the encapsulation is `ETHERNET`.

Verifying LDP Neighbors and Targeted LDP LSPs on Router PE3

Purpose

To verify that Router PE3 has a targeted LDP LSP to Router PE2 and that Router PE3 and Router PE2 are LDP neighbors.

Action

Verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors, using the `show ldp neighbor` command.

```
user@PE2> show ldp neighbor
Address            Interface          Label space ID         Hold time
192.0.2.2            lo0.0              192.0.2.2:0               43
192.0.2.4            lo0.0              192.0.2.4:0               33
```

Meaning

The output shows that Router PE3 has an LDP neighbor with the IPv4 address of `192.0.2.2`. Address 192.0.2.2 is the lo0.0 interface address of Router PE2. The output also shows that the interface used on Router PE3 for the LSP is `lo0.0`. Verifying that the routers are LDP neighbors also verifies that the targeted LSP is established.

Verifying a BGP Peer Session with the Route Reflector on Router PE3

Purpose

To verify that Router PE3 has a peer session established with the route reflector.

Action

Verify that Router PE3 has a peer session established with the route reflector, using the `show bgp summary` command.

```
user@PE2> show bgp summary
Groups: 2 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0            1          1          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.0.2.7              65000       1597       1612       0       1    12:03:21 Establ
  bgp.l2vpn.0: 0/0/0/0
  bgp.l3vpn.0: 1/1/1/0
  L3VPN.inet.0: 1/1/1/0
```

Meaning

The output shows that Router PE3 has a peer session with the router with the IPv4 address of `192.0.2.7`. Address 192.0.2.7 is the lo0.0 interface address of the route reflector. The output also shows that the peer session state is `Establ`, meaning that the session is established.

Verifying the Layer 3 VPN Routes on Router PE3

Purpose

To verify that Router PE3 has Layer 3 VPN routes to Router CE2, Router CE3, and Router CE5.

Action

Verify that Router PE3 has routes to Router CE2, Router CE3, and Router CE5 in the Layer 3 VPN route table, using the `show route table L3VPN.inet.0` command. In this example, `L3VPN` is the name configured for the routing instance.

```
user@PE3> show route table L3VPN.inet.0
L3VPN.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

198.51.100.10/24      *[Direct/0] 11:13:59
                    > via lt-1/1/10.1
198.51.100.11/24      *[Local/0] 11:13:59
                      Local via lt-1/1/10.1
198.51.100.12/24      *[BGP/170] 11:00:41, localpref 100, from 192.0.2.7
                      AS path: I
                    > to 10.10.6.2 via xe-2/1/0.0, Push 16
198.51.100.13/24      *[Direct/0] 11:54:41
                    > via ge-1/0/1.0
198.51.100.1/24      *[Local/0] 11:54:41
                      Local via ge-1/0/1.0
```

Meaning

The output shows that Router PE3 has a route to the IPv4 subnetwork address of `198.51.100.10`. Address 198.51.100.15 is the interface address of Router CE2. The output shows that Router PE3 has a route to the IPv4 subnetwork address of `198.51.100.12`. Address 198.51.100.10 is the interface address of Router CE5. The output shows that Router PE3 has a route to the IPv4 subnetwork address of `198.51.100.13`. Address 198.51.100.6 is the interface address of Router CE3.

Verifying the Layer 2 Circuit Routes on Router PE3

Purpose

To verify that Router PE3 has a route to Router PE2 in the Layer 2 circuit route table.

Action

Verify that Router PE3 has a route to Router PE2 in the Layer 2 circuit route table, using the `show route table l2circuit.0` command.

```
user@PE3> show route table l2circuit.0
192.0.2.2:NoCtrlWord:5:100:Local/96 (1 entry, 1 announced)
        *L2CKT  Preference: 7
                Next hop type: Indirect
                Next-hop reference count: 1
                Next hop type: Router
                Next hop: 10.10.5.1 via xe-2/2/0.0, selected
                Protocol next hop: 192.0.2.2
                Indirect next hop: 8cae0a0 -
                State: <Active Int>
                Local AS: 65000
                Age: 11:16:50   Metric2: 1
                Task: l2 circuit
                Announcement bits (1): 0-LDP
                AS path: I
                VC Label 315264, MTU 1500
```

Meaning

The output shows that Router PE3 has a route to the IPv4 address of `192.0.2.2`. Address 192.0.2.2 is the lo0.0 interface address of Router PE2. Note that the VC label is `315264`. This label is the same as the incoming MPLS label displayed using the `show l2circuit connections` command.

Verifying the MPLS Routes on Router PE3

Purpose

To verify that Router PE3 has a route to Router PE2 in the MPLS route table.

Action

Verify Router PE3 has a route to Router PE2 in the MPLS route table, using the `show route table mpls.0` command.

```
user@PE3> show route table mpls.0
mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
16                 *[VPN/0] 12:22:45
                      to table L3VPN.inet.0, Pop
315184             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315184(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315200             *[LDP/9] 00:03:53, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Swap 625297
                      to 10.10.6.2 via xe-2/1/0.0, Swap 299856
315216             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
315216(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
315232             *[LDP/9] 12:45:06, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315232(S=0)        *[LDP/9] 12:45:06, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315248             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315248(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315264             *[L2CKT/7] 11:11:20
                    > via lt-1/1/10.0, Pop
315312             *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
315312(S=0)        *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
```

```
315328              *[RSVP/7] 11:26:01, metric 1
                     > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
315360              *[RSVP/7] 11:26:01, metric 1
                     > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
316208              *[RSVP/7] 00:03:32, metric 1
                     > to 10.10.6.2 via xe-2/1/0.0, label-switched-path Bypass->10.10.9.1
316208(S=0)         *[RSVP/7] 00:03:32, metric 1
                     > to 10.10.6.2 via xe-2/1/0.0, label-switched-path Bypass->10.10.9.1
lt-1/1/10.0         *[L2CKT/7] 11:11:20, metric2 1
                     > to 10.10.5.1 via xe-2/2/0.0, Push 301488
```

Meaning

The output shows that Router PE3 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE2. Notice that the `301488` label is the same as the outgoing label displayed on Router PE2 using the `show l2circuit connections` command.

Verifying Traffic Flow Between Router CE2 and Router CE3

Purpose

To verify that the CE routers can send and receive traffic across the interconnection.

Action

Verify that Router CE2 can send traffic to and receive traffic from Router CE3 across the interconnection, using the `ping` command.

```
user@CE2>ping 198.51.100.6
PING 198.51.100.6 (198.51.100.6): 56 data bytes
64 bytes from 198.51.100.6: icmp_seq=0 ttl=63 time=0.708 ms
64 bytes from 198.51.100.6: icmp_seq=1 ttl=63 time=0.610 ms
```

Meaning

The output shows that Router CE2 can send an ICMP request to and receive a response from Router CE3 across the interconnection.

Verifying Traffic Flow Between Router CE2 and Router CE5

Purpose

To verify that the CE routers can send and receive traffic across the interconnection.

Action

Verify that Router CE2 can send traffic to and receive traffic from Router CE5 across the interconnection, using the `ping` command.

```
user@CE2>ping 198.51.100.10
PING 198.51.100.10 (198.51.100.10): 56 data bytes
64 bytes from 198.51.100.10: icmp_seq=0 ttl=62 time=0.995 ms
64 bytes from 198.51.100.10: icmp_seq=1 ttl=62 time=1.005 ms
```

Meaning

The output shows that Router CE2 can send an ICMP request to and receive a response from Router CE5 across the interconnection.

9

PART

Configuration Statements and Operational Commands

● Junos CLI Reference Overview | 1397

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also 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
- Operational Commands